Netzgestütztes Musizieren
Network-centric Music Performance (NMP)

Von der
Carl-Friedrich-Gauß-Fakultät
der Technische Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)

genehmigte Dissertation

von
Zefir Kurtisi
geboren am 21.08.1972
in Donje Ljubinje

Eingereicht am:	26.04.2013
Disputation am:	26.11.2013
1. Referent:	Prof. Dr.-Ing. Lars Wolf
2. Referent:	Prof. Dr.-Ing. Carsten Griwodz

2014

Kurzfassung

„How about making distributed live music over the net?“ – „That won't work, because . . . “ – „Let's just try it!“

Dieser Anfang einer Diskussion unter wissenschaftlichen Mitarbeitern an unserem Institut mündete vor einigen Jahren in die Aufgabenstellung für ein Softwarepraktikum mit dem Ziel, das interaktive verteilte Musizieren im Netz zu realisieren. Die Argumente und Zweifel der anfangs skeptischen Fraktion (einschließlich des Autors dieser Arbeit) wurden durch das Ergebnis der Arbeit hinfällig: die Studenten haben den Prototypen eines Systems für netzverteiltes Musizieren nicht nur implementiert, sondern dessen Funktionsfähigkeit höchst eindrucksvoll mit einer Live Darbietung im lokalen Institutsnetz demonstrieren.

Ungläubigkeit mischte sich mit Begeisterung und wich zuletzt der Frage, warum etwas, was von einer kleinen Gruppe enthusiastischer Studenten umgesetzt wurde, nicht schon längst seinen Weg in den computerisierten Alltag gefunden hat. Dieser Fragestellung sind wir am IBR intensiv nachgegangen.

Ziel unserer Forschungsarbeit am NMP (*Network-centric Music Performance*) Projekt war, die Grenzen netzgestützten Musizierens zu erkunden und den Einsatz eines solchen Systems außerhalb lokaler Netze zu realisieren. Die vorliegende Arbeit fasst die relevanten Ergebnisse dieser Tätigkeiten zusammen.

Der erste analytische Teil liefert dabei einen genauen Einblick in die Anforderungen an ein System, welches Musiker transparent verwenden können, um auditiv in Sessions auf virtuellen Bühnen im Netz eintauchen zu können. Wir werden die Latenztoleranz von Musikern als zentralen Faktor kennenlernen, der die Realisierbarkeit von NMP definiert. Er bestimmt einerseits die Verfahren und Techniken, die bei dem Übergang aus dem Labor in das Internet anzuwenden sind. Andererseits gibt er eine obere Schranke für den Anwendungsradius eines NMP-Systems in Weitverkehrsnetzen vor und beantwortet damit unsere einleitende Frage: netzgestütztes Musizieren ist im Internet nur eingeschränkt möglich – eine musikalische Interaktion im von diesem Netz gewohnten globalen Maßstab wird es nicht geben.

Im zweiten Teil dieser Arbeit gehen wir von der Frage *Ob* NMP möglich ist zu dem *Wie* über. Zum geringen Latenzbudget kommen bei der Realisierung die dem Internet innewohnenden Unzulänglichkeiten beim Transport interaktiver Echtzeitdaten hinzu. Die sich daraus ergebenden Herausforderungen werden untersucht und die in unsere NMP-Software implementierten Lösungen vorgestellt. Diese sind aufgrund der ihnen zugrunde liegenden Eigenschaften von Rechnersystemen und Netzen universell gültig, so dass die vorgestellten Ansätze als Blaupause für die Realisierung solcher Systeme herangezogen werden können.

Die aus zahlreichen Untersuchungen unseres NMP-Systems mit vielen Musikern positiven Rückmeldungen belegen, dass interaktives und netzverteiltes Musizieren im Internet unter definierten Bedingungen möglich ist. Damit ist der initiale Diskurs zu Ungunsten der Skeptiker entschieden.

Abstract

„How about making distributed live music over the net?“ – „That won't work, because . . . “ – „Let's just try it!“

These words started a discussion among the research staff at our institute and ended up in a task for a practical course several years ago. The goal of that course was to let a group of students evaluate the feasibility of distributed interactive music making in a network. The arguments and doubts of the initially skeptical researchers (including the author of this work) became obsolete with the outcome of the study: the students did not only implement a system to support networked music performances, but also demonstrated its functionality with a highly impressive live demonstration of a music session held in our institute's LAN.

The first mixture of disbelief and excitement quickly gave way to the question why something that could be realized by a small group of enthusiastic students did not yet found its way into the mainstream. We have investigated this issue at the IBR very detailed within the NMP (*Network-centric Music Performance*) project.

The goal of our NMP project was to evaluate and define the general practicability of network distributed music performances and to realize a system that enables them outside the institute's LAN. The work at hand describes the most relevant results of our research activities in this field.

The first analytical part provides a deep insight into the requirements for a system to provide musicians a transparent access to auditive sessions performed on virtuall stages in the network. We introduce the musicians' latency tolerance as a core factor, that dictates the practicability of NMP. On one hand, it defines methods and techniques to be used for moving from the lab to the Internet. On the other hand, it sets an upper bound for the application radius of NMP in wide area networks and so answers our introductory question: the applicability of network distributed music performances in the Internet is very limited – musical interactions with this network's typical global scope will never work.

In the second part of this work we move from *If* NMP is possible to *How* it can be realized. In addition to the low latency budget we face the general shortcomings of real-time data transmission over the Internet. The resulting challenges are evaluated and the approaches we implemented in our NMP software are discussed. Since the challenges are defined by generic properties of computing systems and networks, they are by principle universally valid. Therefore, the proposed approaches can be used as blueprint for the implementation of NMP-systems.

The overall positive feedback we received during numerous evaluations of our-NMP-system with many musicians proves that network centric music can be performed in the Internet under defined conditions. Thus, the initial discourse was closed in the sceptic's disfavour.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

KAPITEL 1

EINLEITUNG

Das Internet und die damit einhergehende Vernetzung der Menschen weltweit haben einen immensen Einfluss auf das Leben des Einzelnen und auf die Zivilisationen allgemein. Der damit herbeigeführte Umbruch ist gewaltiger als alle bisherig dokumentierten: ein Vergleich mit der Industrialisierung zeigt, dass zwar beides zu einem radikalen Wandel im Arbeitsleben der Menschen geführt hat. Darüber hinaus vollzieht das Leben als solches beim Übertritt in das Internetzeitalter eine Wandlung, die es in diesem Umfang vorher nicht gab. Hier brechen Lokalitäten auf, die „Welt wird zum Dorf"; das Netz wird zur allumfassenden Wissens- und Informationsquelle, man muss es nur bedienen können.

Die Relevanz des einen Netzes wird wirtschaftlich und gesellschaftlich daran sichtbar, dass immer mehr Berufe ohne das Internet nicht mehr denkbar sind. Auch soziologisch gewinnt das Netz an Bedeutung, die aktuelle 'Web-2.0' Euphorie ist nichts anderes als der konsequente Schritt des vernetzten Informationsabrufs hin zum vernetzten Informationsaustausch, die Teilnehmer gehen vermehrt dazu über, Daten nicht nur zu konsumieren, sondern dem globalen Netz hinzuzufügen. Ob diese Entwicklung in eine bessere und gleichberechtigtere Welt führen und welche Nachteile diese Entwicklung mit sich bringen wird, wird sich – mit Blick auf Internet- und Online-Spielsucht, sozialer Verarmung und Verletzung der Persönlichkeitsrechte – erst in naher Zukunft zeigen.

Für die technische Entwicklung ist das Internet nach wie vor eine treibende Kraft, indem sich die Eröffnung neuer Anwendungsmöglichkeiten und die daraus resultierenden höheren Anforderungen an das Netz stetig wechselseitig stimulieren. Der schrittweise Ausbau der Netz-Infrastruktur hat stufenweise die Kapazität und die Stabilität der Verbindungen erhöht, so dass sich das Internet in weniger als 20 Jahren vom Übertragungsmedium für Textnachrichten zu dem *einen Netz* mit globaler und mobiler Omnipräsenz gewandelt hat.

Heute im Jahre 2007 geht die Industrie dazu über, das Internet als alleiniges Übertragungsmedium für alle Daten zu etablieren. Das von der Deutschen Telekom propagierte *Triple Play* soll den Transport aller Inhalte zum Benutzer (in diesem Fall Telefon, Internet und TV-Programm) über eine gemeinsame Leitung über das Internet vereinen. Der dafür erforderliche Ausbau wird mittelfristig dem Benutzer zuhause die gleiche Netzanbindung bieten, wie sie heute in lokalen Netzen vorherrscht, der Flaschenhals der *letzten Meile* entfällt und wird weitere Anwendungsmöglichkeiten erschließen. Denkbar ist die Realisierung der Vision eines Netz-Computers, die erstmals vor der Jahrtausendwende aufkam und die Reduktion des Computers auf ein reines Zugriffsmedium auf die im Netz gehaltenen Daten und Datenverarbeitung beinhaltet, prägnant ist der

Ausdruck *The network is the computer*.

Tendenzen hin zu diesem Szenario sind bereits heute dort bemerkbar, wo auch die verminderte Übertragungskapazität in Privathaushalten die Auslagerung von Daten in das Netz nicht zu stark behindert. So wird EMail oft ausschließlich auf den Servern gespeichert, Web-Kalender verwalten Termine, Fotosammlungen, Videos und Archive werden in das Internet ausgelagert. Wissen wird großteils im Netz aggregiert, die Suchmaschine Google wird bei praktisch jeder Suche fündig und die freie Mitmachenzyklopädie Wikipedia zieht qualitativ und quantitativ mit etablierten Werken mit jahrhundertelanger Tradition gleich.

Neben der Funktion als reines Daten- und Wissensaggregat hat die allgemeine Verfügbarkeit von breitbandigen und pauschal abgerechneten Internetanschlüssen in Privathaushalten in den vergangenen Jahren viele interaktive Anwendungen ermöglicht. Benutzer telefonieren kostenlos miteinander über IP-Telefonie, Netzwerkspiele erfreuen sich immer größerer Beliebtheit, Telearbeiter sind an Projektbesprechungen über Videokonferenzanlagen von daheim mit der Firma verbunden. Dabei ermöglicht die eingesetzte Technik den Menschen, an entfernten Orten präsent zu sein, man spricht von der so genannten Telepräsenz [76].

1.1 Musizieren im Internetzeitalter anno 2004

Die Existenz etablierter Verfahren und Anwendungen, Menschen für unterschiedlichste Zwecke über das Internet virtuell zusammenzuführen, legt nahe, nach Möglichkeiten für das verteilte und netzgestützte Musizieren zu fragen.

Wenn die Technik des Internets reif dafür ist, Menschen per Videotelefonie miteinander zu verbinden und sie sich über Kontinente hinweg unterhalten zu lassen, ist sie auch gut genug, um aktive Musiker miteinander zu verbinden?

Im Jahre 2004, als wir uns am IBR erstmalig mit der Thematik des Musizierens im Internet beschäftigt haben, konnten wir diese Frage größtenteils verneinen. Es gab zwar die ersten Forschungsarbeiten, die sich damit beschäftigten, ob verteiltes Musizieren überhaupt für den Musiker praktikabel ist. Auch existierten erste praktische Versuche, die jedoch stets auf proprietäre Netze aufsetzten und meist einen eingeschränkten Funktionsumfang hatten.

Eine genauere Anforderungsanalyse führte schnell zur Einsicht, dass eine solche Anwendung aufgrund der geringen Latenztoleranz schwer realisierbar ist und in einem nicht deterministischen Netz wie dem Internet nicht zuverlässig funktionieren kann. Dieser Einschränkung bewusst wollten wir zunächst feststellen, ob das verteilte Musizieren im Netz Musikern überhaupt eine akzeptable virtuelle Bühne bieten kann, auf der sie wie im realen Leben spielen können.

Für eine solche Untersuchung wurde im Wintersemester 2004/05 am IBR ein Softwareentwicklungspraktikum durchgeführt mit dem Ziel, einen NMP Prototypen zu entwickeln, der netzgestütztes Musizieren zunächst in einem lokalen Netz ermöglicht.

Das Praktikum wurde erfolgreichen mit einer Live-Vorführung abgeschlossen, die zwar einerseits die Praktikabilität des verteilten Musizierens demonstrierte, gleichzeitig aber auch die Grenzen der Implementierung verdeutlichte. So stieß das System bereits im lokalen Netz an die Grenze der Latenztoleranz, ein Einsatz in größeren Netzen mit höherer Übertragungsverzögerung war damit ausgeschlossen. Dabei konnten auch im lokalen Netz mit nahezu idealen Eigenschaften Paketverluste nicht verhindert werden, so dass keine durchgängig überzeugende Audioqualität gewährleistet werden konnte. Zusätzlich kämpfte der Prototyp mit Instabilitäten und Problemen bei der Langzeitsynchronität.

Die nachgewiesene Machbarkeit netzverteilten Musizierens hat uns zu einer Fortführung der Forschungs- und Entwicklungstätigkeit am NMP-Projekt ermuntert, deren Erkenntnisse in dieser Arbeit zusammengefasst werden.

1.2 Motivation

Die Motivation für die Arbeit an NMP ist vielfältig: vom Standpunkt des Musikers aus erschließt die Bereitstellung eines Werkzeugs zum netzgestützten Musizieren komplett neue Freiheiten und Kommunikationsmöglichkeiten, für die beteiligten Forscher birgt sie die Herausforderung, im Internet eine eigene Klasse neuer Kommunikationsanwendungen zu begründen.

Die Zweckmäßigkeit von NMP als Werkzeug für Musiker wurde im Laufe dieser Arbeit wiederholt aus erster Hand nachgewiesen: Musiker, die zum Testen des Prototypen mit NMP verteilt musiziert haben, haben sich wiederholt nach Einsatz- und Erwerbsmöglichkeiten eines solchen Systems erkundigt. Das Interesse dabei war weniger reine Technikbegeisterung als vielmehr praktischer Natur: die Musiker begannen sofort zu rechnen, wie viel Arbeit, Zeit und Kosten gespart werden könnten, wenn statt Fahrten zum gemeinsamen realen Übungsraum manche Übungen mit Hilfe von NMP virtuell durchgeführt werden könnten.

Der Wegfall der Logistik mag in der heutigen Zeit, die durch die Aktualität von Themen zu Klimaproblematik und CO_2 Neutralität bei der Mobilität geprägt ist, das augenscheinlichste Potenzial eines NMP-Systems darstellen. Tatsächlich können jedoch Möglichkeiten geschaffen werden, die vorher nicht praktikabel waren. Die Basis dafür sind nicht die Kosten oder der Aufwand, sondern die für die Mobilität erforderliche Zeit, die in der virtuellen Welt entfällt. Man denke dabei an dünn besiedelte, aber mit guter Netzinfrastruktur versorgte Gebiete wie Skandinavien oder Australien, in denen der nächstgelegene Musiker Hunderte Kilometer weit entfernt wohnt. Es muss nicht einmal die geografische Distanz sein, die mit einem NMP-System überbrückt werden kann. Ein Interessent aus den USA beschrieb, dass er für das Ein- und Auspacken des Instrumentes und der Fahrt von einem zum anderen Ende von Los Angeles zwei Stunden pro Richtung benötigt – so bleibt sein Saxophon viel zu oft unbenutzt.

Neben diesem Hauptanwendungsgebiet in der Freizeitmusik ist der Einsatz beim E-Learning im institutionalisierten Lernen denkbar: ein Musiklehrer müsste sich nicht auf die Lehrtätigkeit im eigenen Institut beschränken oder lange Fahrten absolvieren. Auf der anderen Seite wären Musikschüler und -studenten nicht auf die Teilnahme an Veranstaltung in der eigenen Bildungseinrichtung beschränkt und hätten die Freiheit, sich mit Schülern und Studenten entfernter Institute zu Ensembles zusammenzuschließen.

Im E-Learning allgemein und für das Musizieren speziell gilt, dass Lernen eine soziale Komponente beinhaltet und somit nicht vollständig in die virtuelle Welt verlagert werden kann. Beim E-Learning hat sich daher die Mischform des Blended Learning besonders bewährt, bei der Präsenzveranstaltungen nur teilweise durch virtuelle Lerneinheiten ersetzt oder um diese ergänzt werden. Auf NMP übertragen würde diese Erfahrung bedeuten, dass das wöchentlich stattfindende Üben im Proberaum nicht ersetzt, sondern durch zusätzliche virtuelle Trainings ergänzt wird.

Die aufgeführten Potenziale eines Systems für netzgestütztes Musizieren mögen ungeschönt einen Eindruck vermitteln, dass die (Musik)Welt nur auf die Realisierung von NMP gewartet hat. Erst ein genauer Blick auf die zugrunde liegende Technologie zeigt, warum – trotz der rasanten Entwicklung von Internet und Computertechnik – diese Umsetzung noch nicht erfolgt ist: netzgestütztes Musizieren stellt Anforderungen an ein solches System wie keine andere Anwendung. Die Anforderungen hinsichtlich Latenz, Interaktivität, Audioqualität, Synchronität und Langzeitstabilität übersteigen die Möglichkeit der heutigen Technik.

Diese Hürde stellt die wesentliche wissenschaftliche Motivation für die Realisierung von NMP dar. Es gilt, die technologischen und physikalischen Barrieren bei der Einhaltung der genannten Anforderungen zu analysieren und daraus verlässliche Angaben über Einsatzmöglichkeiten und Grenzen zu definieren. Darüber hinaus muss die Analyse anschließend praktisch mit einem Prototypen verifiziert werden, der eine Anwendbarkeit des netzgestützten Musizierens unter den ermittelten Bedingungen und definierten Grenzen nachweist.

Ein solcher Nachweis ist nicht auf das netzgestützte Musizieren beschränkt: können die hohen Anforderungen hinsichtlich Interaktivität und Latenz an ein NMP-System erfüllt werden, eröffnen sich Realisationsmöglichkeiten für ähnliche Systeme. Da es in dieser Hinsicht kaum anspruchsvollere Anwendungen gibt – was kann schon interaktiver oder latenzkritischer sein als das gemeinsame Musizieren? – bestimmen wir mit dieser Arbeit die untere Schranke des Machbaren.

Unsere Motivation beruht somit, neben der Verwirklichung einer neuen Kommunikations- und Kollaborationsplattform für Onlinemusik, auf der Festlegung von Möglichkeiten und Verfahren für höchstinteraktive Anwendungen, zusammen mit der Abschätzung des potentiellen Einsatzradius.

1.3 Aufbau der Arbeit

Die Arbeit ist wie folgt gegliedert. Im kommenden Kapitel werden technische Grundlagen aus den Bereichen Netz- und Audiotechnik erörtert, die ein Verständnis der hier betrachteten Problematik ermöglichen sollen.

Bevor wir einen Blick auf verwandte Arbeiten werfen können, stellen wir in Kapitel 3 die Herausforderungen für NMP dar, die sich aus den Anforderungen durch die Musiker einerseits und durch technische Einschränkungen andererseits ergeben. Diese besonderen Schwierigkeiten verdeutlichen die Abgrenzung zu verwandten Arbeiten, die in Kapitel 4 diskutiert werden.

Der zentralen Komponente Latenz ist Kapitel 5 gewidmet, in dem eine Latenzanalyse eines NMP-Systems vorgenommen und basierend darauf die Abschätzung des theoretischen Anwendungsradius durchführt wird.

Die für die ermittelten Herausforderungen von uns gewählten Lösungsansätze werden ausführlich im darauf folgenden Kapitel 6 beschrieben. Die zugrunde liegenden Einschränkungen implizieren strikte Vorgaben an Design und Implementierung unseres NMP-Systems, welche in den darauf folgenden Kapiteln 7 und 8 beschreiben werden.

In Kapitel 9 präsentieren wir die Ergebnisse der Evaluation unseres NMP-Prototyps im realen Einsatz mit Musikern. Wir diskutieren die objektiven Messwerte von in unterschiedlichen Netzen und verschiedenen Musikern durchgeführten Sitzungen und stellen sie den subjektiven Bewertungen der Teilnehmer gegenüber.

Nach dem erbrachten Nachweis der Funktionsfähigkeit von NMP in definierten Umgebungen fassen wir in Kapitel 10 unsere Arbeit zusammen und werfen einen Blick auf aktuelle Entwicklungen der für NMP relevanten Technologien.

KAPITEL 2

GRUNDLAGEN

In diesem Abschnitt werden die zum Verständnis der NMP-Problematik erforderlichen Grundlagen erläutert. Er soll dabei die existierenden Schwierigkeiten bei der Umsetzung von verteiltem Musizieren in einem paketvermittelnden Netz wie dem Internet verdeutlichen und die dabei von uns gewählten Lösungsansätze nachvollziehbar machen.

Wir beginnen mit einem Blick auf das Musizieren in realen Umgebungen, da diese die Erwartungen und Anforderungen der Musiker an NMP definieren.

In einem anschließenden kurzen Abriss über das Internet heben wir die Faktoren hervor, die für das interaktive Musizieren im Netz besonders relevant sind.

Ein letzter Abschnitt stellt potentielle Systemarchitekturen vor und erläutert unseren Ansatz eines zentralistischen Systems mit Bezug auf die angestrebte Funktionalität.

2.1 Musizieren im realen Leben

Wir wollen mit NMP das interaktive Musizieren ins Internet bringen und müssen zunächst auf das Musizieren in natürlicher Umgebung blicken, um die erforderlichen Betriebsparameter für die Realisierung eines solchen Systems aufzustellen.

Als erstes gilt es, aus den verschiedenen Formen des gemeinsamen Musizierens diejenigen Arten zu identifizieren, die unser NMP-System unterstützen soll. Als relevanten Parameter haben wir den Grad der Interaktivität identifiziert, der in realen Szenarien eine hohe Bandbreite hat.

Diese beginnt an einem Extrem mit praktisch nicht-interaktiven Anwendungen wie der professionellen Produktion kommerzieller Musik in Plattenstudios. Die Aufnahme der Stücke erfolgt dabei meist sequenziell in der Form, dass die Musiker einzeln aufgenommen werden. Über Kopfhörer hören sie die bisherige Aufzeichnungen oder den Grundtakt des Musikstücks und spielen bzw. singen dazu ihren Anteil ein, der als separate Tonspur aufgezeichnet wird.

Dieses *Recording* wird mit dem *Mastering* abgeschlossen, bei dem die einzelnen Spuren bearbeitet, skaliert und gemischt werden. Die gesamte Produktion kann dabei vollständig ohne Interaktion erfolgen, d.h., im Extremfall müssen die beteiligten Musiker überhaupt nicht interagieren.

Neben den hohen technischen Anforderungen an die Qualität der verwendeten Geräte stellt die Realisierung eines Musik-Studios als Anwendung im Netz keine technisch hohe Herausforderung dar. Der Austausch der Audiodaten zwischen dem Studio und den beteiligten Musikern erfolgt als Download über gewöhnliche zuverlässige Übertragungswege wie dem FTP oder HTTP. Geeignete Aufnahmegeräte beim Musiker vorausgesetzt, können solche Systeme mit bestehender Technik umgesetzt werden. Entsprechende Varianten sind daher bereits heute in verschiedenen Produkten erhältlich, wie bspw. der *Digital Musician Link* DML von Steinberg (http://www.digitalmusician.net).

Als bereits realisierbare Anwendung zielen wir mit NMP daher nicht auf das Studio-Szenario, unterstützen es gleichwohl mit der Bereitstellung aller während des gemeinsamen Musizierens aufgezeichneten Tonspuren zur nachträglichen Produktion.

Die nächst folgende Stufe auf der Interaktivitätsskala bilden die quasi-interaktiven in Form von Symphonieorchestern oder professionellen Musikkonzerten. Gemeinsames Kennzeichen ist dabei, dass zuvor bis ins kleinste Detail einstudierte Abläufe abgespielt werden. Die Interaktivität beschränkt sich dabei auf die Einhaltung des gemeinsamen Grundtaktes und das Feinjustieren musikalischer Parameter. Diese werden von einer zentralen Stelle vorgegeben, wie bspw. vom Dirigenten, über den eine Interaktion der Musiker untereinander indirekt erfolgt. Im Falle des Orchesters ist eine direkte Interaktion schon aufgrund der räumlichen Ausdehnung eines Ensembles und der damit einhergehenden Ausbreitungsverzögerung des Schalls nicht praktikabel.

Übertragen auf eine netzverteilte Anwendung, stellt dieses quasi-interaktive Szenario eine relativ niedrige technische Herausforderung dar. Die Musiker müssen dabei untereinander durch geeignete Mittel synchronisiert werden, wogegen das Abspielen des gemischten Audiosignals mit einer höheren Latenztoleranz erfolgen kann.

Die Spitze auf dieser Interaktivitätsskala bilden spontane Zusammenkünfte von Musikern, die – ohne vorheriges Einstudieren – ihnen bekannte Stücke in Jam-Sessions spielen oder improvisieren. Hier muss jeder Musiker alle anderen hören, bewerten und sein Spiel kontinuierlich anpassen. Die Interaktion erfolgt in Echtzeit, sowohl in musikalischer als auch in sprachlicher Form. Dieser zweite Faktor ist ein wesentlicher Unterschied zu den bisher genannten Szenarien: zur Improvisation gehört, das gespielte Stück während einer Sitzung etappenweise zu spielen und es kontinuierlich zu kommentieren und zu diskutieren.

Diese Form größter Interaktivität stellt analog die höchsten Ansprüche an die Latenz und damit gleichzeitig an die Realisierbarkeit als netzverteilte Anwendung. Auf dieses Szenario zielen wir mit der Umsetzung von NMP und unterstützen damit implizit auch die erstgenannten, technisch weniger herausfordernden.

Welche Eigenschaften muss NMP nun haben, um solche Szenarien zu ermöglichen? Die Anforderungen haben sich während der Arbeit an NMP zu einem einfachen Kriterium herunterbrechen lassen: möglichst wenig Abweichungen zu dem, das man in natürlicher Umgebung gewöhnt ist. Dazu gehört neben der soeben erwähnten Fähigkeit, sowohl in Form von Musik als auch von Sprache zu interagieren, eine Kommunikation in Echtzeit, um die gewohnte Interaktion nicht zu beeinträchtigen.

Diese qualitative Anforderung müssen wir quantifizieren, um den gültigen Wertebereich für den Parameter Latenztoleranz in NMP zu ermitteln. Blicken wir dazu zunächst auf musikalische Vorgaben, um eine Schranke zu ermitteln. Um auf jede Aktion eines Mitspielers reagieren zu können, muss die Latenz geringer sein als das geringste Aktionsintervall. Als Extremwert wird dabei in der Musik der Trommelwirbel genannt, den Musiker mit einer Rate von bis zu 30 Hz kontrollieren können [8]. Das entspricht, wenn man die Schläge als einzelne Ereignisse wertet, einem Aktionsintervall von 33 ms. Zwar synchronisiert kein Musiker auf einzelne Schläge des Wirbels, Ereignisse wie Start und Ende dessen sind für das Zusammenspiel jedoch sehr wohl wichtige Parameter. Eine Systemlatenz von 30 ms stellt damit die erste plausible Unterstützung für die getroffene Wahl der Latenzschranke.

Die zweite Unterstützung erfahren wir aus der Physik der Schallausbreitung in der Luft. Nehmen wir dazu die Ausbreitungsverzögerung innerhalb einer Sitzung bei der zwei Musiker fünf Meter voneinander entfernt spielen. Die zu einer Aktion eines Musikers vom Mitspieler initiierte Reaktion hört dieser erst nach ca. 30 ms – der Ausbreitungsverzögerung des Schalls auf dem Hin- und Rückweg.

Es kommt also nicht von ungefähr, dass bei unseren Versuchen die Musiker NMP bei einer Verzögerung von 30 ms als transparent anwendbar bewerten – es sind just die Bedingungen, die sie in ähnlicher Art im Hobbykeller oder in der Musikschule antreffen. Für unser NMP-Projekt bedeutet diese Erkenntnis, dass wir das Erreichen dieser Systemlatenz als höchstes Ziel definieren und alle sonstigen Parameter daran ausrichten.

2.2 Kommunikation im Internet

In diesem Abschnitt wollen wir die wesentlichen Grundzüge des Internet darstellen, die den Betrieb eines NMP-Systems bestimmen. Das heutige Internet ist als unverzichtbares Medium im Alltag etabliert. Neben der weiterhin hohen Aktualität im Bereich Forschung und Entwicklung hat es damit auch Einzug in das Allgemeinwissen gehalten. Ein entsprechendes Grundverständnis der Thematik, so wie es bspw. aus dem entsprechenden Artikel zum Thema Internet auf der Online-Enzyklopädie Wikipedia abrufbar ist, wird daher im weiteren Verlauf als bekannt vorausgesetzt. Eine noch tiefere Behandlung, wie sie die zahlreichen RFCs beinhalten, ist für das Verständnis der Problematik nicht unbedingt erforderlich.

Diejenigen Aspekte, die für den Transport zeitkritischer interaktiver Daten über das Internet von besonderer Bedeutung sind, beleuchten wir in diesem Abschnitt etwas genauer.

2.2.1 Aufbau und Eigenschaften

Das Internet ist ein globales Netz, das aus dem Zusammenschluss unzähliger Sub-Netze gebildet wird, welche wiederum ein Zusammenschluss sind, gebildet aus Knoten in Form von Endgeräten bzw. Netzkomponenten und den als Kanten zwischen ihnen angenommenen Verbindungen. Jeder Knoten in diesem globalen Netz ist zu jedem Zeitpunkt durch seine IP-Adresse eindeutig identifizierbar und ermöglicht den Datentransport von und zu jedem beliebigen Knoten. Die zu übertragenden Daten werden beim Sender paketisiert, mit Quell- und Zieladresse versehen und versendet.

Der Aufbau des Internets als Zusammenschluss kleinerer Sub-Netze impliziert eine hierarchische Topologie. Pakete reisen von der Quelle zum Ziel auf diesem Pfad entlang immer bis zu derjenigen Ebene, in der beide Knoten enthalten sind. Die im Internet Protokoll (IP) enthaltenen Mechanismen stellen dabei sicher, dass jeder Knoten weiß, wie ein Paket auf seinem Weg zum Ziel weiterzuleiten ist.

Im Kontext von NMP ist diese vereinfachte Sicht auf das Internet zum Problemverständnis zunächst ausreichend. Wir werden sie an den Stellen direkt ergänzen, in dem zusätzliche Details und Aspekte dem Verständnis zuträglich sind.

2.2.1.1 Übertragungskapazität

Der für den Anwender eines Netzes primär relevante Betriebsparameter ist die Übertragungskapazität, für die üblicherweise auch der aus der Signal- und Analogtechnik übernommene Begriff *Bandbreite* verwendet wird. Er gibt den nominellen Wert der Übertragungsleistung eines Netzes pro Zeiteinheit wieder, wobei heute die Einheiten $kbps$ und $Mbps$ für Kilobit bzw. Megabit pro Sekunde gebräuchlich sind. In Zusammenhang mit lokalen und Kernnetzen trifft man häufiger auch auf den nächsthöheren Präfix $Gbps$ als Angabe für Gigabit pro Sekunde.

Dieser Wert bezieht sich immer auf die Kapazität auf physikalischer Ebene und kann von dem für den Nutzer tatsächlich nutzbaren Wert stark abweichen. In einem auf Fast-Ethernet basierenden Netz kann bspw. unter optimalen Bedingungen ein Wirkungsgrad von 80% erreicht werden, so dass in einem 100 Mbps LAN eine Nutzdatenrate von 80 Mbps bzw. 10 MB/s möglich ist. In einem drahtlosen Netz nach IEEE802.11n mit zwei Strömen kommt man dagegen unter realen Betriebsbedingungen nicht annähernd an die nominale Kapazität von 300 Mbps. Dabei erhöhen die bei der störanfälligen Datenübertragung über Luft erforderlichen Maßnahmen zur Sicherung und Fehlerkorrektur die Nutzdaten beträchtlich. Unter optimalen Bedingungen kann so im Labor ein Wirkungsgrad von knapp 60% erreicht werden, wohingegen in realer Umgebung typischerweise Nettodatenraten von maximale 100 Mbps erreichbar sind.

Verlässt man den Bereich lokaler Netze und betrachtet hierarchische Netzverbunde wie das Internet, trifft man auf dem Netzpfad auf verschiedene Übertragungskapazitäten. Die für den Datenaustausch zwischen zwei Knoten bestimmende ist jedoch immer die kleinste. Bei über kommerzielle Zugangsprovider am Internet angebundene Teilnehmer ist das meist die *Letzte Meile*, also der Abschnitt vom letzten Verteilerknoten im Zugangsnetz zum häuslichen Anschluss.

Bei der kommerziellen Vermarktung von Internetanschlüssen für Privatanwender ist die Bandbreite praktisch das einzige Preis bestimmende Kriterium – die dem Kunden in Rechnung gestellten Beträge richten sich meist ausschließlich nach der Kapazität der *Letzten Meile*. Dementsprechend sind die vermarkteten Angaben wie *ADSL mit 16 Mbps Down und 1 Mbps Up* mit Vorsicht zu genießen. Denn der meist klein gedruckte Hinweis *bis zu* kann sich bisweilen signifikant auf die tatsächlich vorhandene Übertragungskapazität auswirken. Die nominalen Werte beziehen sich auf die Leistungsfähigkeit des zugehörigen Verteilerknotens im Zugangsnetz bei einer vorab bestimmten maximalen Auslastung durch die gleichzeitige Verwendung mehrerer Teilnehmer.

Für die Anwendbarkeit von NMP in solchen Netzen ist die real verfügbare Übertragungskapazität von essenzieller Bedeutung. Diese muss nicht nur einen gegebenen Mindestwert überschreiten, sondern diesen auch kontinuierlich verfügbar halten.

2.2.1.2 Übertragungsverzögerung

Die Übertragungsverzögerung ist die zweite relevante Kenngröße einer Netzverbindung. Sie gibt die mittlere Verzögerung zwischen Sende- und Empfangszeitpunkt eines übermittelten Paketes an.

Dieser Wert setzt sich zusammen aus der physikalischen Ausbreitungsverzögerung elektromagnetischer Wellen im Übertragungsmedium und der Verarbeitungszeit in den Netzknoten zusammen.

Der erste Anteil bestimmt dabei die untere Schranke der Übertragungsverzögerung, die für einen bekannten Netzpfad mit einer Ausbreitungsgeschwindigkeit von etwa $\frac{2}{3}$ der Lichtgeschwindigkeit geschätzt werden kann. Eine Verzögerung von 10 ms kann bei der Signalübertragung über 2000 km Netzdistanz wegen dieser physikalischen Gegebenheit nicht unterschritten werden.

Zusätzlich werden die Pakete auf ihrem Weg vom Sender zum Empfänger in jedem Netzknoten durch die anfallende Verarbeitung verzögert. Unvermeidbar sind dabei für Pufferung entstehende Latenzen, die bestimmte Operationen mit sich bringen. So kann bspw. ein Router ein Paket erst dann weiterleiten, nachdem er die für das Routing benötigte Zieladresse ermittelt hat. Auch die auf Höchstleistung getrimmten Core-Router kommen nicht umhin, vor einer Weiterleitung wenigstens die IP-Header einlesen zu müssen.

Diese verfahrensbedingte Verzögerung steigt mit der Anzahl der an eine Übermittlung beteiligten Netzknoten, wobei der kumulative Wert für einen konstanten Pfad gleich bleibt. Einen variablen Anteil fügen dagegen die Operationen auf den Netzknoten hinzu, die durch Betriebsparameter beeinflusst werden und zu nicht vorhersehbaren Verzögerungen führen. An erster Stelle sind hier Verweilzeiten von Paketen in Warte-

schlangen zu nennen, mit denen Überlastungen der Verarbeitungseinheiten oder Netzverbindungen ausgeglichen werden.

War die Verzögerung noch bis vor wenigen Jahren, in denen sich die Verwendung des Internets auf latenzunkritische Dienste wie Email und Surfen konzentrierte, ein eher nebensächlicher Parameter, hat sich seine Bedeutung mit dem Aufkommen interaktiver Anwendungen wie Onlinespielen oder IP-basierter Telefonie deutlich erhöht.

Beim Ausbau der Netzinfrastruktur wird daher seit geraumer Zeit nicht mehr nur auf die Bereitstellung ausreichender Netzkapazität geachtet, sondern auch viel Wert auf die möglichst verzögerungsarme Verbindung zweier Punkte im Netz gelegt. Dass ein Paket auf seiner Reise zwischen benachbarten Ländern in Europa einen Umweg über die USA und mehreren Satelliten macht, gehört mittlerweile der Vergangenheit an.

Für NMP ist diese unmittelbare Erreichbarkeit von Server und Clients über geringst mögliche Netzdistanzen für einen praktikablen Einsatz essenziell.

2.2.1.3 Varianz der Übertragungsverzögerung (Jitter)

Die zuletzt genannte Variabilität der Verarbeitungszeit von Paketen in den Netzknoten führt zu einer variablen Übertragungsverzögerung auf demselben Netzpfad. Nacheinander von einem Sender zum Empfänger übertragene Pakete können daher verschiedene und zum Teil deutlich voneinander abweichende Laufzeiten haben.

Die Abweichungen entstehen durch unterschiedliche Verweilzeiten der Pakete in den Warteschlangen, die wiederum durch den jeweiligen Füllstand der Queues und die Auslastung der ausgehenden Verbindungen festgelegt wird. Diese nicht vorhersagbare Abhängigkeit bewirkt, dass die Verweilzeiten eines Paketes in einem Netzknoten nicht deterministisch und am ehesten in Form einer stochastischen Verteilung zu beschreiben sind. Diese Verteilung beginnt bei der minimalen Verarbeitungszeit eines Paketes im Optimalfall einer sofortigen Weiterleitung, hat ihr Maximum bei der zum mittleren Füllstand der Queues entsprechenden Wartezeit und schließt auch eine unendliche Verweilzeit für die Fälle ein, dass Pakete aufgrund überlasteter Verbindungen verworfen werden müssen.

Für den gesamten Weg vom Sender zum Empfänger lässt sich die Übertragungsverzögerung folglich als stochastische Dichtefunktion beschreiben, die sich zusammensetzt aus den konstanten Anteilen und der Summe der Dichtefunktionen aller beteiligten Netzknoten.

Im täglichen Gebrauch schätzen wir diese Verteilung mit dem Diagnose Werkzeug `ping` (Packet internet groper) ein. Mittels `ping` sendet der eigene Rechner ICMP-Pakete zu einem Zielrechner, der diese unmittelbar beantwortet. Primär soll damit geprüft werden, ob ein Zielrechner im Netz erreichbar ist, über die gemessene Zeit zwischen Versende- und Empfangszeitpunkt der beteiligten Pakete ist darüber hinaus die Übertragungsverzögerung für beide Richtungen ermittelbar, für die der Begriff Paketumlaufzeit bzw. *RTT* (für Round-Trip-Time) verwendet wird. Eine größere Anzahl sequenziell ermittelter RTT Werte ermöglicht somit den Rückschluss auf die Dichtefunktion der Übertragungsverzögerung zwischen diesen beiden Netzknoten.

In Abbildung 2.1 ist ein solcher `ping` Durchlauf für die Verbindung zwischen dem IBR und dem KIT (Karlsruher Institut für Technologie) dargestellt. Die in der letzten dargestellten Zeile ermittelten Statistiken geben uns bereits einen zusammengefassten Eindruck darüber, dass eine NMP-Session zwischen diesen beiden Instituten aufgrund des hohen Jitters problematisch wird. Die Ermittlung der Dichtefunktion erfordert eine statistische Auswertung von über längere Zeiträume gewonnenen RTT-Messungen.

Eine Dichtefunktion, die auf qualitative Annahmen empirischer Erkenntnisse fußt, ist in Abbildung 2.2 für einen gegebenen Netzpfad dargestellt. Der Wert von RTT_{min} wird bestimmt durch die konstanten Werte von Ausbreitungsverzögerung und minimaler Verarbeitungszeit. Pakete, die auf ihren Weg bei Überlas-

```
PING kit.de (188.93.15.10) 56(84) bytes of data.
64 bytes from 188.93.15.10: icmp_seq=1 ttl=50 time=34.8 ms
64 bytes from 188.93.15.10: icmp_seq=2 ttl=50 time=34.2 ms
64 bytes from 188.93.15.10: icmp_seq=3 ttl=50 time=33.9 ms
64 bytes from 188.93.15.10: icmp_seq=4 ttl=50 time=34.4 ms
[...]
64 bytes from 188.93.15.10: icmp_seq=15 ttl=50 time=38.0 ms
64 bytes from 188.93.15.10: icmp_seq=16 ttl=50 time=26.9 ms
64 bytes from 188.93.15.10: icmp_seq=17 ttl=50 time=35.9 ms
64 bytes from 188.93.15.10: icmp_seq=18 ttl=50 time=26.9 ms
64 bytes from 188.93.15.10: icmp_seq=19 ttl=50 time=26.6 ms
64 bytes from 188.93.15.10: icmp_seq=20 ttl=50 time=36.2 ms
--- kit.de ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 19114ms
rtt min/avg/max/mdev = 26.674/33.693/38.768/3.955 ms
```

Abbildung 2.1: Ping Statistiken zwischen IBR und KIT

tung verworfen werden, haben formal eine unendliche Latenz, so dass im Diagramm der Wert für RTT_{max} fehlt. Im Erwartungswert RTT_{avg} hat die durch die Dichtefunktion über der x-Achse gebildete Fläche ihren Schwerpunkt.

2.2.1.4 Zusammenhang zwischen Kapazität, Verzögerung und Jitter

Aus der Darstellung der ursächlichen Zusammenhänge beim Entstehen variierender Latenzen ist ersichtlich, dass die drei Netzparameter Kapazität, Verzögerung und Jitter untrennbar zusammenhängen. Die Variabilität der Übertragungszeit wird im Wesentlichen von der Verzögerung der Pakete in den Netzknoten auf ihrem Weg

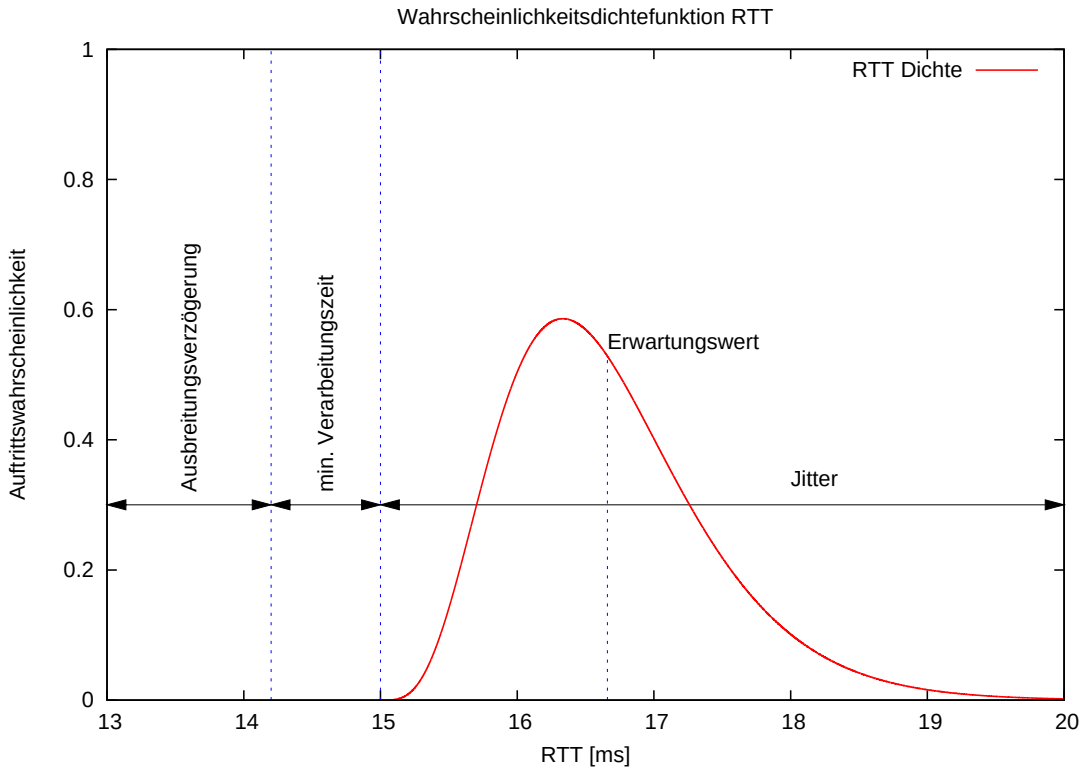

Abbildung 2.2: Dichtefunktion der Übertragungsverzögerung im Netz

vom Sender zum Empfänger bestimmt. Diese Komponente ihrerseits wird bestimmt von der Verweildauer der Daten in den Warteschlangen.

Ein Paket wird immer dann in eine Warteschlange eingefügt, wenn die ausgehende Verbindung durch früher empfangene Pakete belegt ist. Wie schnell ein bestimmter Füllstand der Queues abgebaut wird, hängt unmittelbar von der Kapazität der betreffenden Verbindung ab. Eine höhere Kapazität sorgt für eine schnellere Abarbeitung vorgehaltener Pakete und damit für einen geringeren mittleren Füllstand der Warteschlangen selbst. Eine höhere Kapazität bei gleich bleibender Auslastung verringert somit die Verweildauer von Paketen in einem Netzknoten und reduziert gleichzeitig den Jitter.

Dieser Zusammenhang lässt sich immer dann beobachten, wenn die Netzinfrastruktur bei Erreichen kritischer Auslastungen ausgebaut wird und im Zuge dessen die Kapazität sprunghaft ansteigt. Der Jitter nimmt zunächst merklich ab, um bei weiter steigender Auslastung allmählich wieder zuzunehmen.

Mitunter können Internet- und Netzinfrastrukturanbieter die Reduktion des Jitters als primäres Ziel eines Ausbaus ins Auge fassen, stellt es doch ein effektives und kostengünstiges Mittel, höheren Anforderungen für neue Anwendungen gerecht zu werden.

2.2.1.5 Dienstgüte (QoS)

Die Best-Effort Charakteristik des Internet sorgt für eine hohe Auslastung aller verfügbaren Ressourcen, indem Pakete von jedem Netzknoten unmittelbar und undifferenziert weitergeleitet werden, sobald die ausgehenden Verbindungen verfügbar sind. Für alle nicht-interaktiven Anwendungen ist dieser Ansatz gut geeignet und stellt sicher, dass so viele verschiedene Dienste nebeneinander existieren und zufriedenstellend funktionieren können. Auf diese wirkt sich ein Jitter nur wenig aus: ob die Bilder einer Webseite sofort oder erst nach zwei Sekunden sichtbar sind, spielt keine funktionale Rolle.

Diese Relevanz dreht sich für interaktive Anwendungen, die dadurch gekennzeichnet sind, dass die ausgetauschten Daten nur eine kurze Gültigkeit haben und unmittelbar danach wertlos werden, komplett ins Gegenteil. Zwar legen Kapazität und Ausbreitungsverzögerung primär fest, *ob* eine Anwendung auf diesem Pfad überhaupt durchführbar ist. *Wie* gut sie dann tatsächlich geeignet ist, bestimmt zum größten Teil der Jitter.

Um diesen Anforderungen gerecht zu werden, wurden Mechanismen für die Dienstgüte (*QoS*, Quality of Service) entworfen. Die Grundidee dabei ist die Möglichkeit, Pakete zu klassifizieren und diese mit unterschiedlichen Prioritäten zu bearbeiten. Eine interaktive Anwendung könnte dabei ihre Daten als zeitkritisch markieren und dabei die Gültigkeitsdauer mitgeben.

Die Netzknoten, die dieses Paket bearbeiten, können anhand dieser Angaben entscheiden, ob ein Paket bevorzugt zu behandeln ist oder es u.U. schon so lange unterwegs war, dass es innerhalb der vorgegebenen Gültigkeitsdauer den Empfänger nicht erreichen wird und daher getrost verworfen werden kann.

Für die Anwendung der Dienstgüte kommen zwei unterschiedlich strikte Ansätze zur Anwendung. Das erste ist die Priorisierung der Pakete nach *DiffServ* (Differentiated Services [65]), die die soeben beschriebene bevorzugte Bearbeitung von Paketen anhand der in ihnen enthaltenen Typendeklaration beinhaltet. DiffServ unterstützende Netzknoten müssen in der Lage sein, den entsprechenden Wert im Paket zu identifizieren und es einer geeigneten Verarbeitungseinheit zuzuführen.

Allerdings kann die Priorisierung nur eine relative Verbesserung gegenüber weniger stark priorisierten Anwendungen ermöglichen, eine absolute Garantie für die Einhaltung vorgegebener Parameter kann sie nicht gewähren. Ganz offensichtlich nützt eine Priorisierung dann nichts, wenn der Umfang der entsprechend deklarierten Daten die vorhandene Kapazität übersteigt.

Garantieren kann man die Einhaltung dagegen nur durch Reservierung. Diesen Ansatz verfolgen Verfahren wie das *IntServ* mit der Datenklasse *Guaranteed QoS* [86]. Er beruht darauf, dass für eine gegebene Anwendung auf dem gesamten Netzpfad alle Ressourcen reserviert werden und nicht anderweitig verwendet werden dürfen. Die Netzknoten tauschen die erforderlichen Information über das *RSVP* (Reservation Protocol [96]) aus und belegen die entsprechenden Ressourcen vor.

Der prinzipielle Nachteil der Reservierung liegt in der daraus resultierenden sub-optimalen Auslastung der Ressourcen. Um diese gewähren zu können, müssen diese vorab quantifiziert und angegeben werden. So muss für ein Telefongespräch die maximal zu erwartende Datenrate bekannt sein, ebenso die für eine Videoübertragung. Diese worst-case Allokation verschwendet die verfügbaren Ressourcen daher, da die während Sprechpausen oder weniger dynamischen Videosequenzen geringeren Auslastungen nicht anderweitig verwendet werden können.

In der Netztechnologie ist Dienstgüte schon relativ lange ein Thema – im Internet gibt es diese Funktionalität dagegen nicht. Zum einen liegt das an der praktischen Umsetzbarkeit: damit das Verfahren funktioniert, muss es auf allen Netzknoten entlang eines Pfades unterstützt werden. Bietet auch nur eine Komponente keine Unterstützung dafür, macht QoS nur begrenzt Sinn. Das spiegelt sich im tatsächlichen Einsatz der Technologie wieder. So ist es für die heimische IP-Telefonie hilfreich, VoIP-Pakete am WLAN-Router bevorzugt zu behandeln – ob und wie der Internet-Provider später diese Information in seinem Netz berücksichtigt ist dagegen unbekannt.

Neben der Umsetzbarkeit gibt es technische Gründe, die gegen einen Internet-weiten Einsatz von QoS abzuwägen sind. Eine Erkennung und Verarbeitung der Typenklassifikation ist zuerst rechen- und zeitintensiv, da neben der Zieladresse auch die entsprechenden QoS-Informationen aus dem Header zu interpretieren sind. Die dann fällige Abarbeitung nach Priorisierung oder Reservation ist um ein vielfaches komplexer als das simple Weiterleiten des Paketes. Besonders für Komponenten in Kern-Netzen kann es daher günstiger sein, die Zuverlässigkeit wie oben beschrieben durch die Erhöhung der Kapazität zu steigern, statt dafür durchgängig QoS einzuführen.

Zusammenfassend kann man feststellen, das Mechanismen für Dienstgüte in Netzen existieren, die vollständig unter die Kontrolle einer Institution fallen. Dies kann das private Heimnetz, ein Providernetz, das Netz einer Universität oder das von vornherein mit QoS Unterstützung aufgebaute Internet2 [91] sein.

Für die Arbeit an NMP bleibt dagegen wichtig festzuhalten, dass es QoS für das Internet im Allgemeinen nicht gibt.

2.2.2 Protokolle

Die Kommunikation im Internet basiert auf einem TCP-IP Referenzmodell, auf das eine Vielzahl von Protokollen für verschiedenste Anwendungen und unterschiedliche Anforderungen aufsetzen. Eine tief gehende Diskussion dieser Thematik ist für das Verständnis der NMP-Problematik weder sinnvoll noch nötig. In diesem Abschnitt sollen lediglich die Grundzüge der für NMP in Frage kommenden Protokolle erwähnt werden, um die getroffene Wahl nachvollziehbar zu machen. Dazu gehören die Motivation für die Wahl der eingesetzten Protokolle und eine Darstellung des durch den Protokollstapel verursachten Overheads.

Das TCP-IP Referenzmodell ist ein vereinfachte Abwandlung des ISO/OSI Referenzmodells, welches in sieben Schichten unterteilt ist. Die ISO/OSI Bitübertragungs- und Vermittlungsschicht werden zur Netzzugangsschicht zusammengefasst, am oberen Ende fallen Sitzungs- und Darstellungsschicht mit der Anwendungsschicht zusammen. In Abbildung 2.3 sind diese Modelle schematisch dargestellt, zusammen mit typischen und am häufigsten eingesetzten Protokollen im TCP/IP-Modell.

Basis für jeglichen Internetverkehr ist das Internet Protokoll *IP* in der Vermittlungsschicht, auf dem in der Transportschicht Protokolle zur verbindungslosen und verbindungsorientierten Kommunikation aufsetzen. Darüber liegen unzählige spezifizierte und beliebig viele proprietäre Protokolle in der Anwendungsschicht.

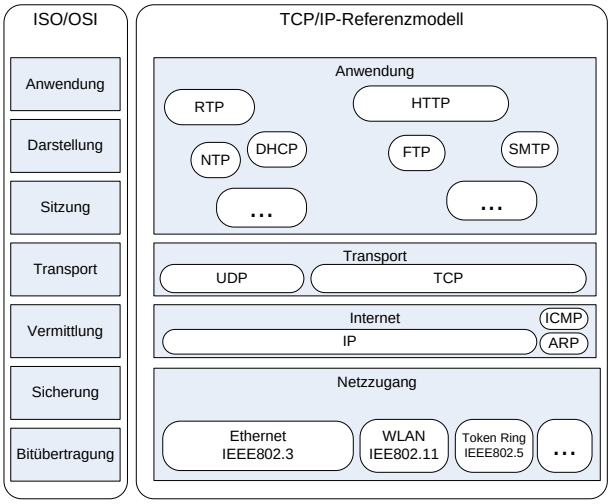

Abbildung 2.3: TCP/IP-Schichtenmodell

2.2.2.1 IP

Das Internet Protokoll ist die Implementierung der Vermittlungsschicht des ISO/OSI-Modells. Da der gesamte Verkehr im Internet über dieses Protokoll übertragen wird, entwickelt sich die Protokollbezeichnung zu einem Synonym für die paketbasierte Datenübertragung im Internet – Begriffe wie IP-Telefonie oder IPTV sind gerade dabei, sich im allgemeinen Sprachgebrauch zu etablieren. Die vierte Version des Protokolls (IPv4 [72]) wurde bereits 1981 definiert. Der IPv4-Header hat eine minimale Länge von 20 Bytes, von denen jeweils 32 Bits für Quell- und Zieladresse verwendet werden. Ein exponentielles Wachstum des Internets und eine zusätzliche Einschränkung des theoretischen Adressraumes von über vier Milliarden Einträgen infolge von Segmentierung bei der Adressvergabe hat zu einer Verknappung von frei verfügbaren IPv4-Adressen geführt. Bei gleich bleibender Entwicklung drohen so die Reserven im Verlaufe des Jahres 2011 auszugehen.

Mit der sechsten Version des Internet Protokolls IPv6 [25] soll dieser Knappheit begegnet werden, indem für die Adressierung eines Endgerätes 128 Bits vorgesehen sind und damit ein praktisch unerschöpflicher Adressraum zur Verfügung steht. Obwohl die ersten Arbeiten für IPv6 bereits 1995 aufgenommen wurden, ist das Protokoll praktisch wenig relevant. So kommt es wegen einer fehlenden durchgehenden Unterstützung von Routern nur selten in lokalen Netzen zum Einsatz. Aus diesem Grund wird im Weiteren nur IPv4 betrachtet, die alleinige Bezeichnung von IP in dieser Arbeit ist als Synonym für IPv4 zu verstehen.

2.2.2.2 TCP

Das TCP (Transport Control Protocol [73]) realisiert auf der Transportschicht eine zuverlässige verbindungsorientierte Kommunikation zwischen zwei Anwendungen. Dabei werden virtuelle Kanäle zwischen Anwendungen über so genannte Sockets aufgebaut und während der Verbindung aufrecht erhalten. Die Zuordnung dieser Kanäle zu den Anwendungen erfolgt über Ports, für die im TCP-Header je 16 Bit für Quell- und Zielport

verwendet werden. Mit der Transportadresse in der Form *IP-Adresse:Port* ist eine eindeutig Zuordnung eines Kanals zu einer Anwendung sicher gestellt.

Nach einem erfolgreichem Verbindungsaufbau kann die Anwendung Bitströme in den virtuellen Kanal schreiben oder von dort lesen. Die Zuverlässigkeit der Datenübertragung wird vollständig vom TCP gewährleistet – das Protokoll muss alle in Abschnitt 2.2.1 genannten Fehlerquellen erkennen und selbständig beheben können. Damit ein Sender Daten nicht schneller versendet, als der Empfänger sie lesen kann, sind Mechanismen zur Flusskontrolle ebenso enthalten, wie Mechanismen zur Überlaststeuerung und Stauvermeidung. Die Absicherung der Datenübermittlung erfolgt über Bestätigungsnachrichten, erfolglose Übertragungsversuche werden wiederholt. Da beim TCP ein Datenstrom und keine Einzelnachrichten übertragen werden, wird auf die Daten sequenziell zugegriffen. Bei einem Paketverlust kommt es so zu einer Blockade, die erst nach erfolgreicher Übermittlung des betreffenden Paketes aufgelöst wird.

Die Zuverlässigkeit der Übertragung wird bei TCP also durch eine erhöhte Latenz erkauft. Für die meisten im Internet verwendeten Dienste ist die Verzögerung nebensächlich, so dass über 80% des Datenaufkommens in Internet über TCP/IP transportiert werden [30].

2.2.2.3 UDP

Das UDP (User Datagram Protocol [71]) ist ein leichtgewichtiges Protokoll auf der Transportschicht, das die Nachrichten-basierte, verbindungslose Datenübertragung zwischen zwei Anwendungen realisiert. Wie bei TCP werden die Ports mit je 16 Bit für Quell- und Zielport adressiert. Darüber hinaus befinden sich im UDP-Header noch je 16 Bit für die Länge des UDP-Paketes und einer Prüfsumme, so dass dieser aus insgesamt acht Byte besteht.

Die in TCP implementierten Verfahren zur Fluss- und Staukontrolle, zur Erkennung und Neuübermittlung verloren gegangener Pakete müssen bei Bedarf von jeder Anwendung selbst implementiert werden.

2.2.2.4 RTP, RTCP, SIP und RTSP

Das verstärkte Aufkommen von multimedialen Echtzeitanwendungen im Internet und der Bedarf von Interoperabilität zwischen verwandten Anwendungen haben zur Spezifikation des RTP (Realtime Transport Protocol [84]) geführt. Es setzt üblicherweise auf UDP auf und wird für die kontinuierliche Übertragung von Medienströmen eingesetzt. Ein minimaler RTP-Header ist 12 Bytes lang und beinhaltet eine 16-Bit Sequenznummer, einen 32-Bit Zeitstempel und eine 32-Bit ID der Synchronisationsquelle (bspw. dem Erzeuger).

Die reine Übertragung der Nutzdaten über RTP wird vom Schwesterprotokoll RTCP (RTP Control Protocol) kontrolliert. Über RTCP werden u.A. Sender- und Empfängerberichte zwischen den Teilnehmern ausgetauscht, anhand derer eine Sicherstellung von Dienstgüteparametern erfolgen kann.

Eine RTP-Sitzung wird von den Teilnehmern selbständig aufrecht erhalten, sobald sie über Signalisierungsprotokolle initiiert wird. Verbreitet sind hierbei für das Streaming das RTSP (Realtime Streaming Protocol [85]), mit dem eine Sitzung zwischen Server und Client aufgebaut und die folgende Datenübertragung kontrolliert werden kann. Das SIP (Session Initiation Protocol [79]) wird verwendet, um eine Sitzung zwischen zwei oder mehr Teilnehmern einzuleiten. RTSP und SIP sind reine Signalisierungsprotokolle und unterliegen damit nicht den gleichen Echtzeitanforderungen wie die Nutzdaten. Sie können daher als Transportprotokoll sowohl UDP als auch TCP verwenden.

2.2.2.5 IP-Multicast

Für Anwendungen, in denen ein Sender identische Daten an mehrere Empfänger versendet, bietet sich das IP-Multicast Verfahren nach [26, 39] an. Es setzt auf das Konzept der Multicast Gruppenadressen auf, zu de-

nen alle Interessenten einer *one-to-many* oder *many-to-many* Kommunikation beitreten. Innerhalb einer solchen Gruppe muss der Sender ein UDP Paket nur einmal senden, die beteiligten Netzwerkknoten (Switches und Router) kümmern sich selbständig um die Vervielfältigung und Verteilung der Pakete an alle Gruppenteilnehmer.

Ideale Einsatzgebiete für Multicast sind daher solche, bei denen jeder Teilnehmer der Gruppe zeitgleich identische Datenströme enthält - man denke dabei an Internetradio oder TV-Streaming.

Es liegt auf der Hand, Multicast als geeignetes Mittel zu erwägen, um die Verteilung der beim NMP-Server gemischten Daten an die angebundenen Clients zu vereinfachen. Statt beim Server über Punkt-zu-Punkt Verbindungen mehrfach Kopien des Mischdatenstroms an jeden Client zu senden, wäre eine einzelne Übermittlung an eine Multicast-Gruppe ausreichend. Dem sinnvollen Einsatz bei NMP stehen jedoch technische und praktische Einschränkung entgegen. Von technischer Seite gibt es zunächst die weiter oben im Zusammenhang mit der Dienstgüte beschriebene Anforderung, dass eine durchgehende Unterstützung für Multicast auf allen Netzknoten des aufspannenden Netzes einer NMP Sitzung erforderlich ist. In der Praxis ist diese bei der Anbindung über kommerzielle Zugangsnetze meist nicht gegeben.

Auch ohne diese Einschränkungen wäre der Einsatz von Multicast mit den in NMP verwendeten Mischverfahren ungeeignet. Wir werden darauf im Abschnitt 6.2 genauer eingehen, im Zusammenhang mit Multicast ist relevant festzustellen, dass die gemischten Pakete sich von Client zu Client unterscheiden. Das betrifft immer die Metadaten (wie bspw. Sequenznummern) und kann bei der Anwendung von positionsbasiertem Mischen auch unterschiedliche Audiodaten einschließen.

Analog zur Dienstgüte bleibt daher auch für Multicast festzustellen, dass es einen potentiell sinnvollen Mechanismus für den Einsatz in NMP darstellt, wegen der Praxisuntauglichkeit für diese Arbeit jedoch nicht weiter betrachtet wird.

2.2.3 Echtzeitdatenströme im Internet

Datenströme sind logisch zusammenhängende Daten, die für die Übertragung im Internet vom Sender paketisiert und sequenziert werden. Anhand der Sequenznummern kann der Empfänger den ursprünglichen Zusammenhang der Daten rekonstruieren. Handelt es sich dabei um eine kontinuierliche Datenübertragung, während derer der Empfänger eingehende Daten unmittelbar verarbeitet, sprechen wir vom *Streaming*. Es findet Verwendung bei unidirektionaler Übertragung audiovisueller Medien, die wenig Interaktion erfordern und bei denen die Verzögerung zwischen dem Senden und dem Abspielen der Daten von nachrangiger Bedeutung ist. Vertreter dieser Kategorie sind Web-Radios oder TV-Übertragungen im Internet, die nach dem Initiieren der Übertragung nur eine schwache nutzerseitige Interaktion ermöglichen. Solche Streaming-Anwendungen können auf TCP aufsetzen und von der zuverlässigen Übertragung profitieren: da keine eigenen Maßnahmen zur Fehlerkorrektur implementiert werden müssen, können die Daten im gleichen Format versendet werden, in dem sie für den lokalen Bedarf vorliegen. Da TCP Stau- und Flusskontrolle enthält und die erneute Übertragung verloren gegangener Pakete die Kontinuität der Datenübertragung beeinträchtigt, muss beim Empfänger mit einem Abspielpuffer gearbeitet werden. Die Länge dieses Puffers wird vor der Initiierung einer Sitzung in Abhängigkeit von der Leitungsqualität bestimmt und kann mehrere Sekunden betragen.

Interaktive Anwendungen haben eine weitaus strikte Anforderung an die Latenz. In Telefonie- und Konferenzanwendungen gilt eine Ende-zu-Ende Verzögerung von 200 ms als Richtwert, bis zu dem eine störungsfreie Sprachkommunikation möglich ist. Um diese Forderung einzuhalten, muss die durch Paketisierung entstehende Verzögerung gering ausfallen. Sie liegt üblicherweise um eine Größenordnung unter der tolerierbaren Latenz, so dass beispielsweise bei IP-Telefonieanwendungen Audiopakete mit konstanter Abspieldauer

von 10-20 ms verwendet werden.

Aufgrund der geringen Latenztoleranz werden in solchen Anwendungen bei der Übertragung verloren gegangene Pakete nicht erneut angefordert, weil die erforderliche Zeit für die Erkennung eines Verlustes, die Neuanforderung und die Sendewiederholung meist die Gültigkeitsdauer des betreffenden Paketes übersteigt. Zusätzlich zu den bei der Übermittlung verloren gegangene Paketen müssen auch beim Empfänger nach ihrem Abspielzeitpunkt eintreffende Pakete als Paketverluste gewertet werden, da sie für das Abspielen nicht mehr relevant sind. Ähnlich wie beim Streaming muss beim Empfänger ein De-Jitter Puffer eingerichtet werden, um Schwankungen der Übertragungsverzögerungen im Netz zu kompensieren. Interaktive Anwendungen müssen bei der Dimensionierung dieses Puffers stets einen Kompromiss zwischen der tolerierbaren Latenz und Paketverlustrate eingehen.

Da bei interaktiven Anwendungen Paketverluste erwartet werden, muss das Übertragungsverfahren robust ausgelegt und der Einfluss verlorener Datenpakete auf die Wahrnehmung des Benutzers minimiert werden. Um Robustheit zu gewährleisten, müssen Abhängigkeiten der Pakete untereinander vermieden werden. Ist dies prinzipiell nicht möglich, muss das Verfahren die Abhängigkeiten erkennen und die zu einem verlorenen Paket gehörende Paketgruppe vollständig verwerfen. Dies wird deutlich bei der Videocodierung, die für die Datenreduktion die Ähnlichkeit benachbarter Bilder anhand von Referenz- und Differenzbildern nutzt und daher zu starken Abhängigkeiten zwischen Paketen einer Gruppe führt. Beim Verlust eines Paketes können dabei davon abhängige Pakete nicht decodiert und müssen verworfen werden. Diese unvermeidbaren Fehler müssen abschließend mit geeigneten Strategien verdeckt werden, um vom Benutzer so wenig störend wie möglich empfunden zu werden. Die Wahl geeigneter Maßnahmen ist dabei anwendungs- und medienspezifisch zu treffen. So kann es bei der Videoübertragung sinnvoll sein, über die eingetretenen Paketverluste hinaus gültige Daten zu verwerfen, wenn sich beispielsweise ein nur teilweise decodiertes Bild auf die Wahrnehmung schlechter auswirkt als das Einfrieren der Darstellung bis zum nächsten vollständigen Bild.

Die Einhaltung geringer Latenz erfordert eine hohen Paketrate, um die Verzögerung durch Pufferung gering zu halten. Abhängig vom übertragenen Medientyp variiert die Nutzdatenrate und damit die Datenmenge, die pro Paket übertragen wird. Da die unterschiedlichen Übertragungsprotokolle auf jeder Schicht den Nutzdaten einen Header hinzufügen, sinkt die Effizienz der Übertragung mit kleineren Paketgrößen. Für eine Abschätzung der für die Übertragung einer Nutzdatenmenge erforderlichen Bandbreite, wird der minimale, durch Protokoll-Header erzeugte, Overhead benötigt. Die Notwendigkeit auf UDP als Transportprotokoll zu setzen impliziert das Einfügen eines UDP- und IP-Headers in jedem Paket. Für die Identifizierung von Absender und Medientyp und die Sequenzierung der Pakete sind ergänzende Headerdaten erforderlich, wie sie beispielsweise bei RTP verwendet werden. Unabhängig davon, ob für eine Anwendung RTP verwendet oder ein eigenes Protokoll auf Anwenderebene eingesetzt wird, sind die genannten Daten jedem Paket anzuhängen, damit ein Datenstrom identifiziert und verarbeitet werden kann. Als grober Richtwert für den Protokoll-Overhead eines Echtzeitdatenstroms lässt sich daher die Datenmenge pro Paket angeben, die für RTP, UDP und IP-Header verwendet wird, die sich zu 40 Bytes pro Paket aufsummieren. Dieser Wert schränkt die Wahl der Paketgröße bei der Übertragung von Datenströmen nach unten dadurch ein, dass der Anteil übertragener Nutzdaten bei ansteigender Paketfrequenz abnimmt.

2.3 Systemarchitektur

Kollaborative Anwendungen im Netz wie NMP basieren auf dem Grundprinzip, dass die relevanten Daten jedes Teilnehmers allen anderen übermittelt werden. Dafür sind die prinzipiell unterschiedlichen Architekturen Peer-to-Peer (P2P) und der klassische Client-Server Ansatz wählbar.

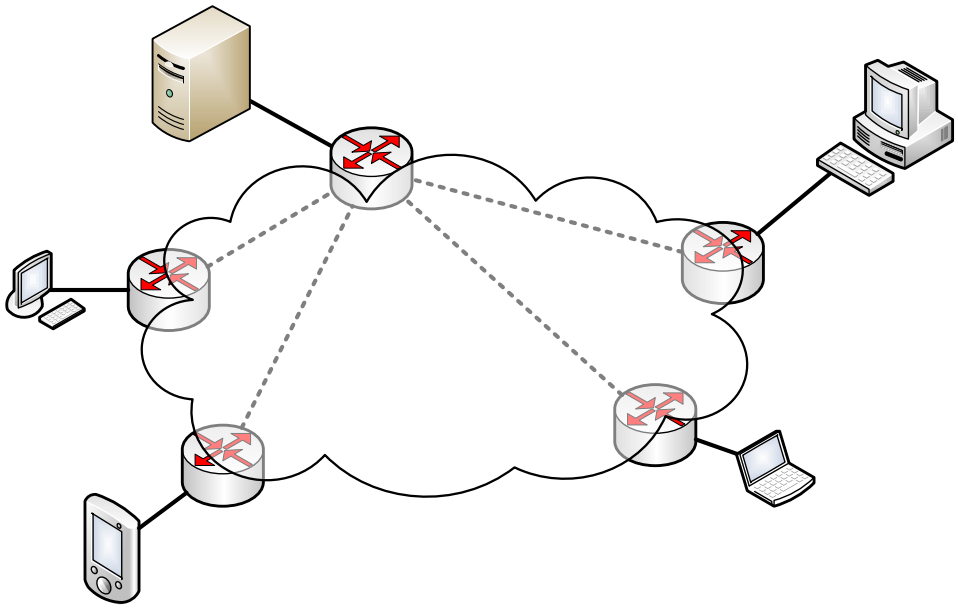

Abbildung 2.4: NMP in einem Client-Server Aufbau

2.3.1 Server-zentriert

Kern eines Client-Server-Systems bildet der Server, der einen wohl definierten Dienst anbietet und diesen über ein wohl definiertes Protokoll den Clients zugänglich macht. Dieses Prinzip ist das älteste und auch heute noch meist verbreitetste bei Netzanwendungen, die aus betriebswirtschaftlichen oder technischen Gründen zentralistisch aufgebaut sind.

Für die Bereitstellung eines Dienstes muss im Server die entsprechende Anwendung laufen, die Anfragen der Clients akzeptiert und beantwortet. Client-seitig muss eine Anwendung eingesetzt werden, die über eine Benutzerschnittstelle den Zugriff zum entsprechenden Dienst bereitstellt. Als Beispiel für solche Client-Server Anwendungen können aus dem Alltagsgebrauch der Zugriff auf EMail oder der Abruf von Webseiten genannt werden.

Für solche grundlegenden Dienste sind die Kommunikationsprotokolle offen bzw. standardisiert und ermöglichen so die Verwendung unterschiedlichster Anwendungen auf Client- und Serverseite. Genügt beispielsweise ein Web-Server dem HTT-Protokoll, können beliebige Web-Browser über HTTP von diesem Inhalte abrufen. Alle heute etablierten Client-Server Anwendungen konnten sich praktisch nur aufgrund der offen gelegten Protokolle durchsetzen. Auf proprietäre Varianten stößt man nur bei sehr speziellen und abgeschlossenen Diensten, wie beispielsweise bei Gültigkeitsabfrage von Kreditkarten beim bargeldlosen Verkehr.

Charakteristisch für eine Client-Server Architektur ist der funktionale Unterschied zwischen den Endsystemen. Der vom Server bereitgestellte Dienst kann dabei von vielen Clients abgerufen werden und ermöglicht so den Zugriff auf Funktionen oder Inhalte. In seiner Urform als Terminal-Host System ermöglichte diese Architektur den gemeinsamen Zugriff vieler Benutzer auf die Rechenleistung eines zentralen Großrechners, heute liegt der Fokus von Client-Server Anwendungen auf den Zugriff auf Inhalte. Dabei spielen Web und EMail die dominante Rolle, die um neue multimediale und interaktive Formen wie Video-Streaming und Online-Spiele ergänzt werden.

Die Realisierung von NMP über einen zentralen Server wie in Abbildung 2.4 dargestellt bietet sich an, da vergleichbare interaktive Anwendungen wie Konferenzen oder Online-Spiele meist diese Architektur ver-

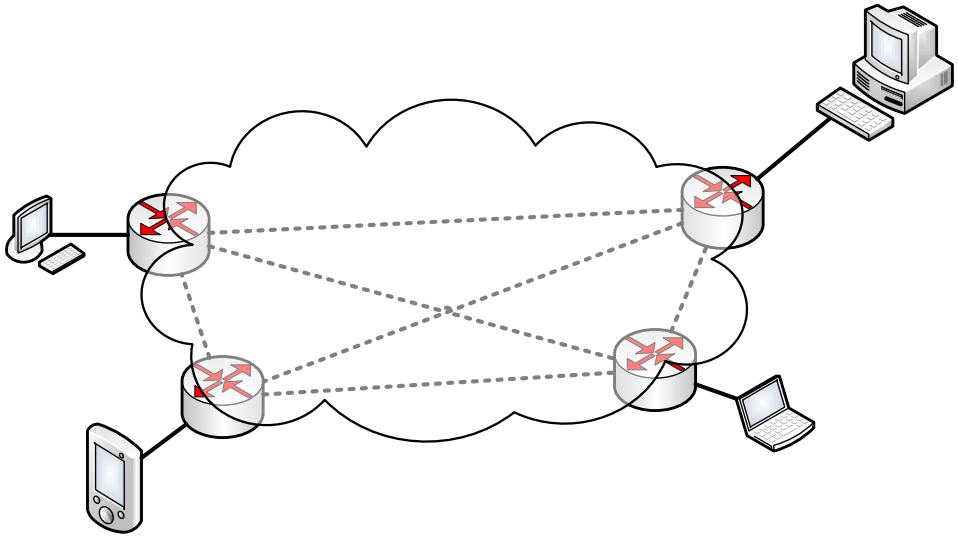

Abbildung 2.5: NMP in einem P2P-Netz

wenden. Ein NMP-Server muss als grundlegenden Dienst das Zusammenführen mehrerer NMP-Clients für das gemeinsame Musizieren in Sitzungen unterstützen und dabei die Kernfunktionen für das Mischen und Synchronisieren der Audiodatenströme bereitstellen. Auf Basis dieses elementaren Dienstes können beliebige ergänzende Dienste aufgebaut und bereitgestellt werden, wie bspw. das Aufzeichnen und zur Verfügung stellen der Audiodaten oder der Ersatz realer Musiker durch virtuelle.

2.3.2 P2P-Ansatz

Im Gegensatz zur Client-Server Architektur kommt ein Peer-to-Peer Netz ohne zentrale Komponente aus. Alle Knoten übernehmen gleichzeitig die Client- und Serverfunktionen und sind dabei gleichberechtigt. Durch den Verzicht auf eine dedizierte zentrale Komponente kann der in einem P2P-Netz bereitgestellte Dienst nicht durch die Entfernung einzelner Rechner unterbunden werden. Diese Robustheit durch das Fehlen der *Single Point Of Failure (SPOF)* hat P2P Anwendungen erst zu dem Durchbruch in (teils illegalen) Internet-Tauschbörsen verholfen. War es bis dahin Inhalteanbietern möglich, Web- oder FTP-Server zu schließen, auf denen urheberrechtlich geschützte oder illegale Medien verbreitet wurden, ist dies in den P2P-Netzen von *Gnutella*, *eMule* oder *BitTorrent* nicht mehr möglich.

Neben der Robustheit gerät in jüngster Zeit die gemeinsame Ressourcenverteilung über alle Knoten beim P2P immer stärker in den Fokus. So verlassen derzeit verschiedene Ansätze für IPTV das Teststadium, in denen die Videodaten vom zentralen Server über P2P an die Teilnehmer verteilt werden. Jeder Rechner ist dabei gleichzeitig selbst Datenserver, der die zuletzt empfangenen Daten zwischenspeichert und bei Bedarf an andere Benutzer weiterleitet. So werden der zentrale Server entlastet und die Netze geschont.

Möglich ist dies nur, wenn jeder Knoten dem P2P-Netz einen Teil seiner Ressourcen zur Verfügung stellt. Im Falle von IPTV-Angeboten wie *Joost* oder *Zattoo* muss jeder Benutzer Festplattenplatz, Rechen- und Netzkapazität bei der Teilnahme bereitstellen.

Eine redundante Aufteilung der Last auf alle Knoten ermöglicht einen höchst dynamischen Aufbau des Netzes und prädestiniert P2P für verteilte Anwendungen, die ohne zentrale Kontrollinstanz auskommen und eine kontinuierliche Fluktuation der teilnehmenden Knoten erfordern.

Prinzipiell ist auch die P2P Architektur für den Einsatz von interaktiven Anwendungen wie NMP denkbar. Sehr erfolgreich wird das von der VoIP Anwendung Skype umgesetzt, die Internet-Telefonie über ihr P2P

Relevanz	Eigenschaft	Peer-to-Peer	Client-Server
funktional	Netzlast [Datenströme]	C·(C-1)	C
	CPU Last	C	1…C
	Einheitliches Mischen	✗	✓
	Synchronität	✗	✓
optional	Virtuelle Bühne	✗	✓
	Recording	✗	✓
	On-Demand Modus	✗	✓
nicht relevant	Robustheit gegen SPOF	✓	✗
	Dynamische Netztopologie	✓	✗
	Ressourcenaufteilung	✓	✗

C = Anzahl NMP-Clients; SPOF = *Single Point Of Failure*

Tabelle 2.1: Für NMP relevante Gegenüberstellung von P2P und Client-Server

Overlay-Netz anbieten. Dabei werden die Audiodatenströme zwischen den Gesprächspartnern über angemeldete Knoten weitergeleitet. Sollen wie beim gemeinsamen Musizieren die Daten aller Teilnehmer mit geringer Verzögerung gemischt werden, muss beim P2P jeder Client mit jedem anderen verbunden sein, womit ein voll vermaschtes Netz entsteht, wie in Abbildung 2.5 dargestellt. Jeder Knoten muss seinen Audiodatenstrom an alle Teilnehmer senden und im Gegenzug die von allen Teilnehmern empfangenen Audiodatenströme mischen und synchronisieren.

2.3.3 Bewertung und Wahl der Architektur

Für die Wahl der Client-Server Architektur für unser NMP ist ausschlaggebend, dass viele Funktionalitäten eine zentrale Instanz erfordern und die Vorteile, die ein P2P Aufbau mitbringt, nicht sinnvoll verwertet werden können.

In Tabelle 2.1 sind die wichtigsten Charakteristiken der beiden Architekturen aufgeführt und in Relevanzklassen für funktionale, optionale und nicht relevante Bedeutung in einem NMP-System gegliedert. In die höchste Kategorie fällt zunächst einmal die Belastung des Netzes, da die zu übertragenden Datenmengen erheblich sind. In einem voll vermaschten Netz muss jeder Client seine Audiodaten jedem anderen Client zukommen lassen und im Gegenzug die Datenströme von allen anderen empfangen und verarbeiten. Besonders beim Einsatz in Zugangsnetzen mit asymmetrischer Bandbreitenverteilung für Sende- und Empfangsrichtung ist der zentralistische Ansatz vorzuziehen. Auch dort, wo genug Übertragungskapazität vorhanden ist, käme der Einsatz eines P2P basierten NMP wegen der quadratischen Abhängigkeit der zu übertragenden Daten von der Anzahl der Teilnehmer schnell an Auslastungsgrenzen.

Bei der Betrachtung der CPU Lasten in beiden Szenarien ist zu berücksichtigen, dass beim Client-Server Ansatz durch das Konzentrieren aller Datenströme Verarbeitungsmöglichkeiten entstehen, die in einem verteilten System nicht umsetzbar sind. Das betrifft bspw. das Mischen der Audiodaten zu einem gemeinsamen Mischsignal, das in einem P2P Netz in allen Knoten durchzuführen ist. Demgegenüber muss der NMP-Server den Mix nur einmal erzeugen und diesen an alle Teilnehmer versenden. Dadurch kann die Bilanz in diesem Punkt für den zentralistischen Ansatz um Faktor C (Anzahl Clients) günstiger ausfallen. Werden dagegen virtuelle Bühnen simuliert, in denen für jeden Musiker individueller Raumklang gemischt wird, erhöht sich der Rechenaufwand auf den Wert, wie er im P2P Fall entstehen würde.

Neben dem nachteiligen Ressourcenbedarf bei P2P können für NMP erforderliche Kernfunktionen nicht oder nur sehr beschränkt dezentral umgesetzt werden. Intuitiv sichtbar wird dies beim Mischen und Synchronisieren der Audiodatenströme. Nur ein zentraler Mixer kann gewährleisten, dass alle Teilnehmer ein

durchgängig synchronisiertes Mischsignal ausgeben. Müsste sich jeder NMP-Client individuell um die Synchronisation der eingehenden Datenströme kümmern, können Abweichungen entstehen, so dass nicht jeder Teilnehmer das Gleiche hört. In Abschnitt 6.7 werden wir auf die enormen Schwierigkeiten bei der Synchronisation in NMP eingehen – in einem P2P Aufbau sind diese ungleich höher.

Die Erzeugung des Mischsignals ist ebenfalls eine Aufgabe, die zentral am effektivsten gelöst werden kann. Der Aufbau in realen Musikszenarien verdeutlicht dies anschaulich. Dort werden die Audiosignale aller Teilnehmer im zentralen Mixer gesammelt und nach dem Mischen ausgegeben. Die Parametrisierung und Verstärkung der Audiopegel zueinander erfolgt durch die Musiker vor dem Training oder durch Tonmeister in professioneller Umgebung. Dieser *Soundcheck* gelingt nur, wenn alle Musiker das gleiche Mischsignal hören und eine Übereinstimmung bei der Parametrisierung des Mixers besteht. Ein dezentrales Mischen innerhalb eines P2P basierten NMP-Systems ermöglicht gemeinsames Musizieren nur dann, wenn alle Mixer identische Parameter verwenden. Die Koordination dieser Parametersynchronität ist praktisch – insbesondere für den Soundcheck – nicht umsetzbar.

Die Vorzüge einer zentralistischen Architektur gegenüber dem P2P Ansatz beschränken sich nicht auf diese Kernfunktionalitäten. Auch die meisten übrigen Aufgaben und Möglichkeiten, die sich aus der Wiederverwendung aufgezeichneter Audiodaten ergeben, sind nur mit einem zentralen NMP-Server sinnvoll machbar. Das Sammeln und Speichern individueller und gemischter Audiodatenströme erfordert eine Infrastruktur für breitbandige und speicherplatzintensive Archivierung der Daten und deren Abruf, die nicht an jedem NMP-Client verfügbar ist.

Die Vorteile einer P2P Architektur erweisen sich für NMP als nicht relevant. Sitzungen für das gemeinsame Musizieren sind statisch, die Musiker treffen üblicherweise zusammen und nehmen anschließend daran durchgehend von Sitzungsbeginn bis zum Ende teil. Die strikten Anforderungen an die Betriebsparameter können währen dieser Zeit nur dann gewährleistet werden, wenn die Netztopologie gleich bleibt. Der Ausfall eines Knotens führt damit automatisch zu einem Abbruch der begonnenen Sitzung. Die Robustheit gegenüber SPOF ist so beim NMP nicht relevant, auch besteht kein Bedarf danach, Sitzungen dynamisch aufzubauen. Die wesentlichen inhärenten Vorteile der P2P Architektur kommen bei NMP nicht zur Geltung, die Verwendung dieses Ansatzes ist somit wenig sinnvoll.

Die genannten Nachteile des P2P Ansatzes für NMP entfallen nur bei einem direkten Zusammenspiel von zwei Musikern. Auch dann ist ein praktischer Ansatz ohne zentralen Server nur dann sinnvoll, wenn auf die genannten On-Demand Funktionalitäten verzichtet werden kann. Ein solches P2P-basierendes System wird beispielsweise in [18] an der *isnm* in Lübeck verfolgt.

KAPITEL 3

HERAUSFORDERUNGEN

Wir haben einleitend erläutert, dass die technischen Anforderungen bei der Realisierung eines Systems für das netzverteilte Musizieren ungleich höher sind, als für vergleichbare interaktive kollaborative Anwendungen. Bevor wir im nächsten Abschnitt einen Überblick über den Stand der Technik und verwandter Arbeiten geben, sollen hier diese besonderen Anforderungen an ein NMP-System benannt und die damit verbundenen Herausforderungen erläutert werden.

Wir wollen zunächst einen Blick auf reale Musik-Szenarien werfen und basierend darauf die Erwartungen der Musiker an ein NMP-System festhalten. Ausgehend von diesen Anforderungen identifizieren wir die technischen Herausforderungen und beschreiben die damit verbundenen Schwierigkeiten bei der Realisierung eines solchen Systems.

3.1 Überführung realer Musik-Szenarien in den virtuellen Raum

Basis unserer Arbeit ist das reale Musizieren in einer physikalischen Umgebung, die wir mit unserem System durch die Bereitstellung einer virtuellen Bühne erweitern wollen.

Grundlage dieser Erweiterung ist die Überführung der akustischen Signal- in eine paketbasierte Datenübertragung. Für den Anwender unseres NMP-Systems muss dieser Ersatz des Übertragungsweges möglichst transparent sein, das auditive Empfinden darf sich nicht zu stark von dem in einer realen Umgebung unterscheiden, um eine intuitive Bedienbarkeit des Systems zu gewährleisten.

Da wir uns auf die Übertragung der akustischen Informationen beschränken, können mit unserem System Szenarien nicht unterstützt werden, die zusätzlich auf visuellen Eindrücken oder anderen Hilfsmitteln beruhen. Als Beispiele für solche nicht realisierbaren musikalischen Aufbauten seien Orchester genannt, die einen Dirigenten erfordern, instrumentelle Musik, bei der die sichtbare Motorik beim Spielen wesentlich zur Synchronität beiträgt oder professionelle Musik, die auf der Bühne nur mit Hilfe von Monitor-Lautsprechern durchführbar ist.

Damit sind die für den Einsatz von NMP prädestinierten Szenarien die bereits beschriebene Lern- und Unterhaltungsmusik, die von Anfängern und fortgeschrittenen Anwendern gespielt wird, die keine perfekte Einsatzumgebung erwarten oder systembedingte Einschränkungen tolerieren können.

Ausgehend von einer geringen Abweichungstoleranz von NMP gegenüber dem realen Musizieren, be-

trachten wir im Folgenden, welche Eigenschaften der Anwender von einem NMP-System erwartet und welche technischen Schwierigkeiten die Bereitstellung dieser Eigenschaften mit sich bringt.

3.2 Latenz

Die Gesamtverzögerung, die der Musiker zwischen Erzeugung und Wahrnehmung eines Tones empfindet, ist ein zentraler Aspekt dieser Arbeit und wird als Latenz bezeichnet. Synonym für diesen Ausdruck wird auch die Verzögerung verwendet, wenn Zusammenhänge unterstrichen werden sollen, treten sie in zusammengesetzter Form z.B. als Systemlatenz, Ende-zu-Ende-Verzögerung oder Gesamtverzögerung auf.

Alle beschreiben die Kernanforderung der Musiker, möglichst ohne bewusst wahrnehmbare Pausen zwischen Klangerzeugung und Klangwahrnehmung musizieren zu wollen. Intuitiv erschließt sich die Bedeutung dieser Größe innerhalb eines interaktiven und kontinuierlichen Vorgangs des Musizierens, die ab einer gewissen Höhe die Synchronität und das Tempo des Spiels beeinträchtigt oder es ganz verhindert.

Einen ersten Eindruck über die Größenordnung dieses Wertes gewinnt man beim Blick auf die physikalischen Gegebenheiten in der realen Welt. Hier gilt die Schallgeschwindigkeit in Luft von 343 $\frac{m}{s}$ (bei Raumtemperatur 20°C, s. [95]), ein Audiosignal benötigt also für die Ausbreitung über einen Meter knapp 2.9 ms. Vom Anschlagen einer Gitarrensaite bis zur Wahrnehmung des Klangs vergehen so etwa 3 ms. Ein gemeinsames Spielen mehrere Musiker gelingt nur, wenn eine Synchronität eingehalten wird. Damit ein Spieler A die Reaktion eines Spielers B auf seine Aktion wahrnimmt, muss der Schall die Entfernung zwischen A und B zweimal zurücklegen. Sind die beiden Musiker während einer Übung fünf Meter auseinander, vergehen mindestens 29 ms, bis ein Musiker das Feedback auf seine Aktion wahrnimmt. Musiker sind also in der Lage, diese natürlichen Latenzen zu kompensieren und sich zu synchronisieren, solange die Toleranzgrenze nicht überschritten wird. Dieses ist beispielsweise in einem Orchester der Fall, das aufgrund der großen räumlichen Distanzen der Musiker untereinander nur über die visuelle Synchronisierung durch den Dirigenten gemeinsam spielen kann.

Einen allgemeingültigen Wert für diese Grenze gibt es nicht, da er von unterschiedlichen Gegebenheiten und subjektiven Eigenschaften abhängt. Als Richtwert aus der Literatur wie in [19] oder [83] und eigenen Untersuchungen in [34] können wir einen Bereich der Latenz um 30 ms ausmachen, bis zu dem gemeinsames Musizieren netzgestützt und ohne zusätzliche Hilfsmittel möglich ist. Darüber hinaus bleibt ein NMP-System bis zu einer Gesamtverzögerung von 50 ms bedienbar, wobei die Musiker meist zur Einhaltung des Tempos ein Metronom benötigen. Höhere Latenzen erlauben kein intuitives Bedienen des Systems, ein entsprechendes Training ist jedoch durchaus geeignet, die maximal tolerierbare Verzögerung zu erhöhen.

Die Realisierung eines funktionalen und von Musikern akzeptierten Werkzeugs für die netzgestützte Kollaboration unter Einhaltung dieser maximalen Verzögerung als Zielvorgabe verdeutlicht im Vergleich zu ähnlichen Anwendungen, dass NMP eine eigene Anwendungsklasse darstellt. Stellt man einem solchen System beispielsweise die Internettelefonie mit einer Latenztoleranz von bis zu 300 ms gegenüber, hat man es mit einer um eine Größenordnung striktere Anforderung zu tun. Berücksichtigt man, dass es für beide Anwendungen vergleichbare Funktionsteile (wie z.B. die Audio-Ein- und Ausgabe) gibt, die in beiden Fällen die gleiche absolute Latenz verursachen, steigt der Aufwand, um die genannte Grenze einzuhalten, überproportional an.

Dieser Kernproblematik widmen wir uns im Kapitel 5, in dem wir die möglichen Verzögerungen in einem NMP-System analysieren und eine Abschätzung des Einsatzradius von NMP vornehmen werden.

Über die bloße Einhaltung vorgegebener minimaler Latenzgrenzen hinaus ergeben sich diverse Schwierigkeiten bei der Realisierung von NMP, da der Einsatz bewährter Techniken und Verfahren erschwert oder verhindert wird.

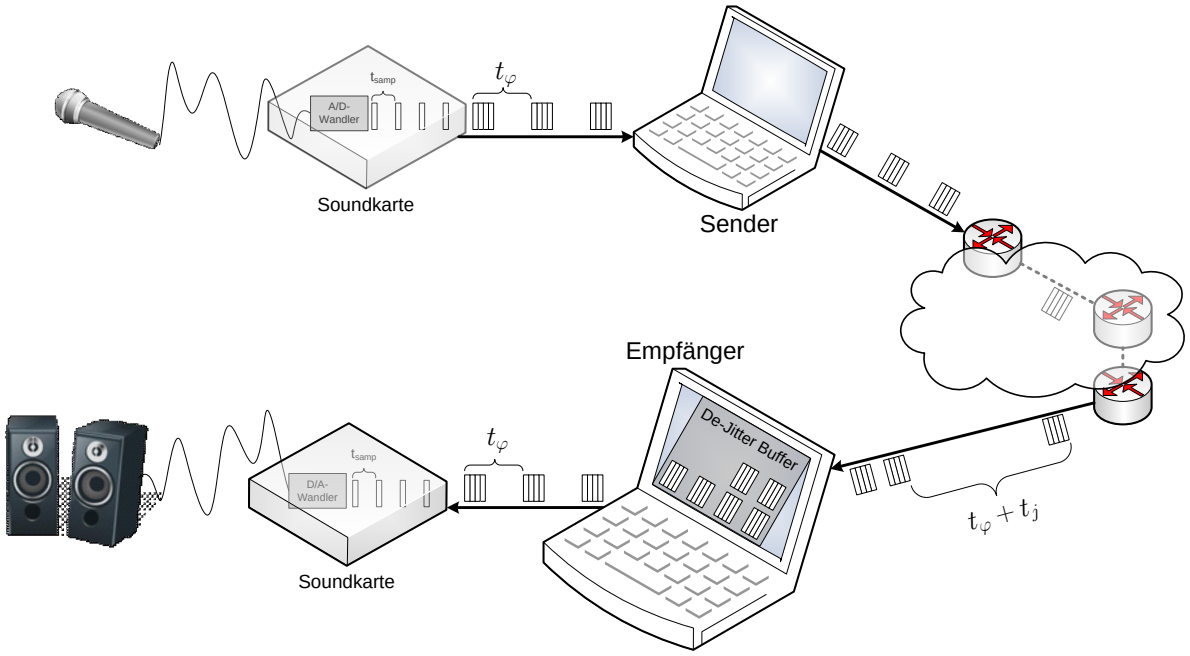

Abbildung 3.1: Übertragung isochroner Daten zwischen Erzeuger und Verbraucher

3.3 Übertragung isochroner Daten über paketvermittelnde Netze

Die Audiodatenströme im NMP haben isochronen Charakter, d.h., es sind Audiosamples in zeitlich konstanten Abständen kontinuierlich von Client zum Server und zurück zu versenden und am Client wieder mit gleicher konstanter Abtastrate auszugeben.

Der Transport solcher Daten über paketbasierte Netze wie dem Internet wird dabei durch verschiedene Aspekte behindert. So führt die erforderliche Serialisierung und Deserialisierung der Audiodaten für die Übermittlung zu einer Verzögerung, die der strikten Anforderungen an geringer Latenz zuwiderläuft.

Weit stärker wiegt für NMP die nicht-deterministische Eigenschaft der Übertragungsverzögerungen in IP-Netzen, die sich in variablen Paketlaufzeiten und Paketverlusten auswirkt. In interaktiven Anwendungen wird dieser Problematik nach Möglichkeit durch den Einsatz von garantierter Dienstgüte (QoS) begegnet, wie er in abgeschlossenen Netzen (bspw. die Kernnetze der Mobilfunkbetreiber) verwendet wird. Andernfalls sind die Anwendungen so auszulegen, dass Paketverluste erkannt und entweder korrigiert oder effektiv verdeckt werden. Um die variierenden Übertragungszeiten zu kompensieren, durchlaufen die Audiodaten nach dem Empfang ein De-Jitter Puffer, wodurch eine zusätzliche Verzögerung verursacht wird – dem kritischen Faktor in einem NMP-System.

3.4 Robustheit gegenüber Paketverlusten

Die Länge des De-Jitter Puffers beeinflusst die Wahrscheinlichkeit, dass ein Paket nach der entsprechenden Wartezeit beim Sender eintrifft. Pakete, die nach dem Abspielzeitpunkt am Empfänger eintreffen, müssen als Paketverluste gewertet werden. In IP-Netzen treten nicht nur Variationen der Übertragungszeiten auf, es können prinzipbedingt auch Pakete auf der Route vom Sender zum Empfänger verworfen werden.

In interaktiven Anwendungen, in denen verlorene Pakete nicht erneut versendet werden, muss daher immer mit Paketverlusten gerechnet werden. Beim NMP sind die Längen der De-Jitter Puffer durch die ein-

zuhaltende Gesamtlatenz sehr stark beschränkt, sodass mit einer beträchtlichen Paketverlustrate gerechnet werden muss.

Etablierte Verfahren zur Vorwärtsfehlerkorrektur und zur Verdeckung von Lücken im Datenstrom sind aufgrund der kurzen Puffer nur sehr eingeschränkt möglich und erfordern spezifische Lösungsansätze. Diesen wichtigen Aspekt werden wir genauer in Abschnitt 6.3 behandeln.

3.5 Audioqualität und Audiokompression

In einer Musikanwendung ist naturgemäß die Anforderung an Audioqualität sehr hoch. Diese Anforderung geht einher mit einer großen Datenmenge für die zu transportierenden und speichernden Audiodaten. Zu bewältigen sind diese nur mit einer durch Audiodatenkompression umzusetzenden Datenreduktion. Dieses trifft im Prinzip für alle Computeranwendungen im Audiobereich zu. Tatsächlich wird höchst selten mit unkomprimierten Audiodaten gearbeitet.

Für die unterschiedlichen Anwendungen haben sich diverse Audiocodecs etabliert, die sich in zwei Klassen unterteilen lassen:

Codecs für Sprache sind echtzeitfähig und werden in interaktiven Anwendungen wie in der Sprachkommunikation verwendet. Sie sind auf die Einhaltung geringer Latenzen optimiert und bieten eine moderate und für Sprachanwendungen ausreichende Audioqualität bei guter Kompressionsrate. Solche Codecs werden in interaktiven Anwendungen wie Mobilfunk, VoIP oder Videokonferenzen eingesetzt, bieten bei Verwendung in der Musik dagegen nur ungenügende Qualität und sind für NMP ungeeignet.

Codecs für Musik sind die in der Öffentlichkeit eher als solche bekannten Verfahren wie bspw. MP3, in denen Audiodaten archiviert und in Anwendungen wie Streaming verwendet werden. Diese Codecs bieten mittlerweile transparente (d.h., vom unkomprimierten Audio durch das menschlichen Gehör nicht unterscheidbare) Audioqualität bei einer Kompressionsrate von 1:10. Diese wird im Wesentlichen dadurch erreicht, indem für den Menschen nicht wahrnehmbare Informationen verlustbehaftet aus dem Audiosignal entfernt werden. Die zugehörige Analyse erfordert die Bearbeitung in großen Datenblöcken und verursacht erhebliche Verzögerungen beim Codieren und Decodieren, die die Verwendung der effizienten Audiocodecs in interaktiven Anwendungen verhindert.

Für weniger zeitkritische Anwendungen existieren auf geringe Latenz optimierte Ansätze, die hohe Effizienz der Musikcodecs auch für interaktive Anwendungen nutzbar zu machen. Für NMP erweisen sich jedoch auch diese als ungeeignet – wir können somit auf keines der etablierten Verfahren zurückgreifen.

Die Untersuchung existierender Verfahren auf Echtzeitfähigkeit und effektiver Datenreduktion bei hoher Audioqualität bildet deshalb einen zentralen Aspekt der Arbeit an NMP. Bei der durchzuführenden Evaluation stellt sich zusätzlich die Wahl geeigneter Metriken für die Beurteilung der Audioqualität als Herausforderung dar, da existierende Verfahren für NMP nur eine unzureichende Aussagekraft haben. Diesen Problemen werden wir uns etwas genauer im Abschnitt 6.4 widmen.

3.6 Kontinuität und Synchronität

Die Kontinuität der Audiodatenströme ist keine NMP spezifische Herausforderung und betrifft auch vergleichbare Anwendungen wie Internettelefonie. Der grundlegende Unterschied von NMP ist die hierbei erforderliche strikte Synchronität aller Teilnehmer untereinander, die den erwähnten Anwendungen fehlt. So werden beim VoIP die Audiodaten der beiden Gesprächspartner unidirektional übertragen, technisch betrachtet findet Streaming statt. In unserem NMP-System müssen die Audiodatenströme aller Teilnehmer einer Sitzung vom Server kontinuierlich und über lange Zeiträume sehr genau synchronisiert werden. Eine zeitliche

Drift der Datenströme untereinander würde vom Musiker eine permanente Anpassung zur Resynchronisierung erfordern und so die intuitive Bedienung des Systems verhindern.

Angesichts der großen Teilnehmerzahl beim Musizieren sind die möglichen Quellen für Asynchronitäten während einer Sitzung sehr hoch, sei es zwischen den Uhren in den Soundkarten und dem Computer jedes Clients oder zwischen den einzelnen Komponenten untereinander. Erschwert wird die Detektion und Behandlung von Asynchronitäten dadurch, dass sich die Driften dynamisch verändern und von Paketverlusten verfälscht werden können. Dieses diffizile Thema greifen wir in den Abschnitten 6.6 und 6.7 auf.

3.7 Mischen von Audio zur Simulation virtueller Bühnen

Einen signifikanten Unterschied von NMP zu vergleichbaren interaktiven Anwendungen ist die Notwendigkeit, die simultanen Audiodatenströme aller Teilnehmer zu einem gemeinsamen zu mischen. Dieses findet in der hier erforderlichen Form bei keiner etablierten Anwendung statt, auch wenn bspw. die Namensgebung bei Audio- oder Videokonferenzanwendungen darauf schließen lässt, dass hier Audiodaten gemischt werden, um den Eindruck von gleichzeitig stattfindender Kommunikation zu erzeugen. Tatsächlich wird in solchen Applikationen meist ein aktiver Kanal bestimmt – dies kann bspw. der lauteste Sprecher sein – und an alle Teilnehmer übermittelt. Ein gleichzeitiges und womöglich unkoordiniertes Durcheinanderreden ist in gesprochener Kommunikation ohnehin nicht sinnvoll, beim NMP ist sie essenziell.

Beim Mischen von Audiodaten bzw. allgemein Audiosignalen gerät man schnell in schwer lösbare Problembereiche aus der wissenschaftlichen und handwerklichen Akustik. Eine generelle Schwierigkeit stellt dabei das Berechnen des digitalen Mix aus den Einzelsignalen dar, welches aufgrund des diskreten Wertebereichs der Abtastwerte zum Übersteuern bzw. zur Reduktion des Signal-Rausch-Abstandes (SNR) im gemischten Signal führen kann. Geläufige Verfahren aus der professionellen Studio-Technik können dieses Problem nur dann beheben, wenn die zu mischenden Audiospuren vollständig vorliegen, um eine Vorab-Skalierung vorzunehmen. Infolgedessen sind diese beim NMP nicht verwendbar, stattdessen müssen geeignete Verfahren entwickelt werden, die eine sinnvolle Abwägung zwischen Übersteuerungsschutz und hohem SNR ermöglichen und den Mixer kontinuierlich parametrisieren.

Eine Vorgabe für unser NMP ist, den Musikern eine virtuelle auditive Bühne zu geben. Das wird dadurch erreicht, dass die einzelnen Audiospuren zu einem mehrkanaligen Mischsignal mit Raumklang kombiniert werden. Abhängig von der Position auf der virtuellen Bühne fließen die Einzelspuren dabei mit unterschiedlicher Gewichtung in die Kanäle des Mischsignals ein und ermöglichen eine akustische Lokalisation jedes Teilnehmers im virtuellen Raum. Diese kann in Abhängigkeit von der verfügbaren Rechenzeit in einem breiten Bereich für die Komplexität erfolgen und bewegt sich zwischen einer statischen Bühne äquidistant platzierter Musiker und gemeinsamer Wahrnehmung und einer Bühne mit beweglichen Teilnehmern, die jedem Musiker ein individuelles und seiner Position entsprechendes Audiosignal bietet.

Zusätzlich zu diesen Problemstellungen aus dem Bereich des Audio-Engineerings sind die Schwierigkeiten zu berücksichtigen, die sich aus der fehlerbehafteten Datenübertragung ergeben. Die Verfahren zum Mischen der Audiodaten müssen robust gegenüber Lücken im Datenstrom sein und bereits beim Entwurf so ausgelegt werden, diese Fehlstellen für den Benutzer möglichst unhörbar zu überbrücken. Diese Problematik behandeln wir im Abschnitt 6.2.

3.8 Zusammenfassung

Die in diesem Abschnitt beschriebenen wichtigsten Herausforderungen bei der Umsetzung von NMP zeigen in erster Linie eines: wir haben es mit einer großen Anzahl von gegenläufigen Problemen zu tun, die sich gegenseitig beeinflussen und die so abzuwägen sind, dass die für netzgestütztes Musizieren geltenden Anforderungen erfüllt werden. Dabei sind wir mit wohlbekannten Schwierigkeiten konfrontiert, genauso wie mit Anforderungen, die nur für NMP gelten. Das beginnt mit der für alle zeitkritischen und interaktiven Netzanwendungen – wie lange soll auf die Daten gewartet werden – und geht über unzählige weitere und teilweise NMP-spezifische Gegenläufigkeiten. Aus den oben aufgeführten Einzelproblemen nennen wir an dieser Stelle beispielhaft folgende zur Verdeutlichung:

Die Wartezeit im Netz bestimmt das Verhältnis von Verzögerung und Paketverlustrate. Je länger auf ein Paket gewartet werden kann, desto höher ist die Wahrscheinlichkeit, es innerhalb dieser Zeit zu empfangen. Kann nur eine kurze Zeit gewartet werden, muss die zu erwartende hohe Verlustrate kompensiert werden, daneben sind viele Verfahren nicht durchführbar, wenn die Wartezeit stark eingeschränkt ist.

Die Paketgröße der Audiopakete bestimmt die Granularität der Anwendung. Sie sollte möglichst klein sein, um die Verzögerung für die Paketisierung gering zu halten. Sie sollte groß sein, um den Header-Overhead der Netzprotokolle zu begrenzen.

Datenredundanz soll eine Rekonstruktion verlorener Datenpakete ermöglichen. Auf der anderen Seite erhöht sie das Datenaufkommen und verursacht zusätzliche Verzögerung, da für die Rekonstruktion die Pufferung mehrerer Pakete nötig ist.

Audiodatenkompression verringert die Datenrate der Audioströme und entlastet Netze und Speichermedien. Gleichzeitig erhöht sie die Verzögerung durch zusätzliche Pufferung und schafft Abhängigkeiten zwischen aufeinanderfolgenden Audioblöcken, die sich im Falle von Paketverlusten höchst nachteilig auswirken.

Die algorithmische Komplexität bestimmt, wie umfangreich die Audiodaten bearbeitet werden können. Auf der einen Seite sollen teilweise aufwendige Operationen auf den Daten ausgeführt werden (Audiocodierung, Mischen, Transformation, usw.), auf der anderen muss der Server in der Lage sein, mehrere Sitzungen mit jeweils mehreren Teilnehmern in Echtzeit zu bearbeiten.

Wir werden in den folgenden Kapiteln diese Punkte genauer untersuchen und geeignete Maßnahmen zur Lösung der Gegenläufigkeiten vorstellen. Wir werden erkennen, dass die gegebenen Anforderungen und Einschränkungen den Spielraum für die einzelnen Parameter stark einschränken und dadurch einen engen Bereich vorgeben, in denen NMP praktikabel umsetzbar ist. Neben den einzelnen technischen Herausforderungen erweist sich so die Balance zwischen den verschiedenen Anforderungen als weitere zentrale Schwierigkeit.

Dieser Einblick in die besonderen Herausforderungen beim NMP ermöglicht uns nun, im nächsten Kapitel einen Blick auf die verwandten Themen und existierenden Forschungsarbeiten zu werfen und uns von diesen abzugrenzen.

KAPITEL 4

VERWANDTE ARBEITEN

Das gemeinsame netzgestützte Musizieren teilt mit vielen ähnlichen Anwendungen die Charakteristiken kollaborativer Echtzeitumgebungen. In den letzten Jahren hat sich, gestützt durch die technologische Entwicklung, auf diesem Gebiet sehr viel getan und dazu geführt, dass diese Systeme bereits im alltäglichen Leben anzutreffen sind. Darunter fallen Anwendungen wie die paketbasierte Sprachkommunikation über IP-Netze (VoIP), Audio- und Videokonferenzen oder das auch hierzulande an Bedeutung gewinnende Spielen im Netz. Ebenso mangelt es nicht an Ansätzen für das Musizieren im Netz. Erst eine genauere Untersuchung dieser Anwendungen offenbart, dass unterschiedliche Problemstellungen bestehen und die eingesetzten Lösungsmechanismen nicht auf das NMP-Szenario übertragbar sind.

In diesen Abschnitt betrachten wir zunächst artverwandte und etablierte Verfahren und diskutieren die generellen Unterschiede zu NMP. Anschließend geben wir einen Überblick über Arbeiten im Bereich netzgestütztes Musizieren und erläutern die Besonderheiten unseres Ansatzes.

4.1 Artverwandte Anwendungen

Von den heute etablierten Kommunikationsanwendungen im Internet soll anhand von wenigen Beispielen verdeutlicht werden, wodurch sich unser NMP-System von ihnen grundlegend unterscheidet. Als Beispiele werden aus dem Audiobereich aufgeführt die Internettelefonie via VoIP als Punkt-zu-Punkt Anwendung und die Client-Server basierte Audiokonferenz mit mehr als zwei Teilnehmern. Daneben werden Online-Spiele als Vertreter von kollaborativen, höchst interaktiven und zeitkritischen Anwendungen betrachtet.

4.1.1 VoIP

Die paketvermittelte Sprachübertragung über das Internet per VoIP (Voice-over-IP) ist mittlerweile allgegenwärtig und erfreut sich wachsender Beliebtheit. Diese *Internet-Telefonie* wurde anfangs nur im engen Kreis technisch begeisterter Teilnehmer verwendet, um die textbasierte Echtzeitkommunikation *Chat* zu ersetzen oder bei Fernbeziehungen günstige Gespräche zu Familienangehörigen oder Freunden zu ermöglichen. Musste man sich zu dieser Zeit noch mit minderwertiger Sprachqualität und umständlicher Handhabung zufriedengeben, hat der Ausbau der Netze und der allgemein höheren verfügbaren Übertragungskapazität die Entwicklung von Produkten rund um VoIP begünstigt und diese Technologie mittlerweile in unserem

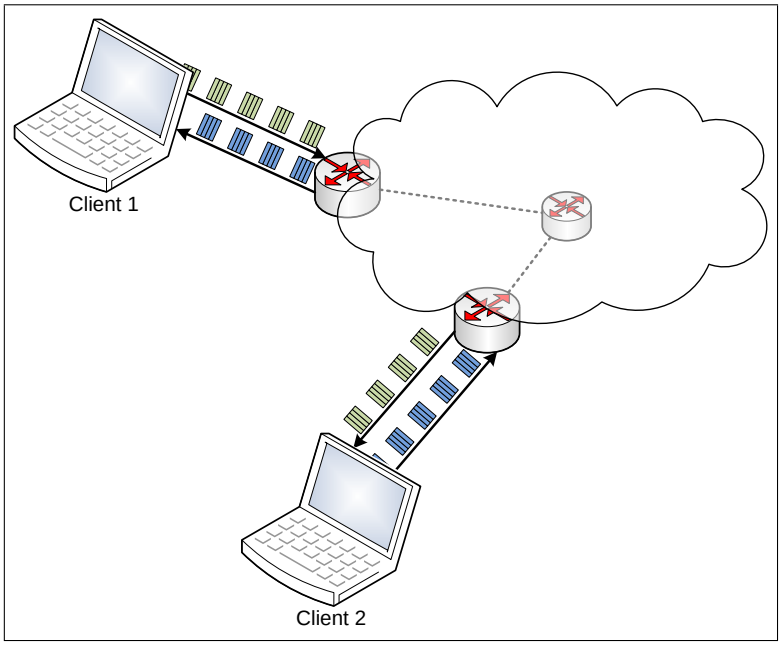

Abbildung 4.1: Prinzip der Internet-Telefonie (VoIP)

Alltag etabliert. Als relevante Treiber dieser Entwicklung seien hierbei beispielsweise die Software Skype[1] genannt, die allen Teilnehmern kostenlose Gespräche untereinander erlaubt, oder die Integration von VoIP-Funktionen in immer mehr DSL-Router für den heimischen Internet-Anschluss. Die damit ermöglichte transparente Unterstützung von Anrufen sowohl über das Festnetz als auch über Internet ermöglicht die Nutzung von VoIP auch für den Laien. Wie signifikant diese Entwicklung fortgeschritten ist, lässt sich daran ablesen, dass der Vorstandsvorsitzende der Deutschen Telekom jüngst die Internet-Telefonie als größte Gefahr für die künftige Umsatzentwicklung des Konzerns bezeichnet.

Mit NMP hat VoIP praktisch nur gemeinsam, dass beides interaktive Audioanwendungen sind, die Audiodaten möglichst verzögerungsarm paketbasiert über IP-Netze übertragen. Wie in Abbildung 4.1 dargestellt, werden die Audiodaten von Teilnehmer A abgegriffen, paketisiert, über das Internet zu Teilnehmer B gesendet, serialisiert und dort in der richtigen Reihenfolge ausgegeben. Gleiches gilt für die Gegenrichtung, sodass aus technischer Sicht Audiodaten auf zwei unabhängigen Verbindungen zwischen den beiden Teilnehmern ausgetauscht werden.

Ein genauerer Blick auf die Abläufe belegt, dass die Gemeinsamkeiten zwischen VoIP und NMP bereits direkt nach dem Abgreifen der Audiodaten von der Soundkarte enden: VoIP kann auf etablierte Sprachcodecs zurückgreifen und versendet die Audiopakete direkt zum Empfänger. Dort angekommen können sie vor dem Abspielen in lange Abspielpuffer vorgehalten werden, um Übertragungsschwankungen im Netz auszugleichen oder verlorene Pakete effizient zu verdecken.

So stellt die Internet-Telefonie praktisch ein gleichzeitiges Audiostreaming zwischen den beiden Gesprächspartnern dar.

Grundlegender Unterschied zum NMP ist das Fehlen von Gleichzeitigkeit und somit die Umsetzung ohne Synchronisations- und Mischerfunktionalität eines zentralen Servers. Das ist in der Sprachkommunikation auch nicht erforderliche, da diese üblicherweise nicht gleichzeitig stattfindet – in jedem sinnvollen Gespräch redet immer nur eine Person. Die Entscheidung, wer diese Person gerade ist, wird den Teilnehmern

[1] http://www.skype.com

überlassen. Um ihnen dieses wie gewohnt ermöglichen zu können, darf eine maximale zeitliche Verzögerung zwischen der Wiedergabe eines Tones und der Wahrnehmung beim Empfänger nicht überschritten werden. Die ITU gibt dafür in der Empfehlung G.114 [48] einen Toleranzwert von 150 ms an, bei dem die Telefonqualität als sehr gut empfunden wird. Bei höheren Verzögerungen wird eine gewohnte Sprachkommunikation erschwert.

Liegen die Gesprächspartner geografisch zu weit auseinander, kann dieser Wert nicht mehr eingehalten werden. Es kommt zu deutlich wahrnehmbaren Verzögerungen, die den Redefluss beträchtlich stören. Diese Einschränkung ist nicht VoIP spezifisch, sie kann auch beim gewöhnlichen Telefonieren beobachtet werden. Insbesondere bei über Satellit geführten Transatlantikgesprächen erschwert die hohe Latenz eine intuitive Gesprächsführung, weil auf Interaktionen nur stark verzögert eingegangen werden kann.

Diese physikalische Hürde gilt für NMP mit der deutlich geringeren Latenztoleranz analog im stärkeren Maße, wir werden in Kapitel 5 näher darauf eingehen.

Ansonsten sind die Unterschiede von VoIP zu NMP gravierend: die deutlich höhere Latenztoleranz ermöglicht es, die Datenübertragung als Streaming zu realisieren. Der kollaborative Aspekt, der durch das Mischen und Synchronisieren der Audiodaten in einem zentralen Server erfolgt, fehlt vollständig. Vom technischen Standpunkt ist VoIP daher eine wenig herausfordernde Anwendung.

4.1.2 Internet-Audiokonferenzen

Das Fehlen kollaborativer Elemente bei der Punkt-zu-Punkt Verbindung zweier Gesprächspartner beim VoIP scheint bei Audio- oder Videokonferenzsystemen auf den ersten Blick enthalten zu sein. Tatsächlich ist es gar nicht oder nicht in dem Umfang verfügbar, der für NMP erforderlich ist.

Unsere kritische Anforderung Latenztoleranz ist bei Konferenzanwendungen sogar noch entspannter als beim VoIP, da hier mit höheren Verzögerungen gearbeitet werden kann. Das ergibt sich aus der unterschiedlichen Zuteilung des Rederechtes in beiden Kommunikationsformen: während es bei einem Zweiergespräch praktisch pausenlos zwischen den Gesprächspartnern wechselt, wird eine Konferenz auch im realen Leben meist mit Moderation oder einvernehmlichen Regeln zur Zuteilung des Rederechtes geführt. Die Interaktivität in Frage-Antwort Dialogen ist so ungleich höher als bei in Vortragsstil gehaltenen Konferenzen.

Vom Aufbau her ähneln Konferenzanwendungen NMP. Wie in Abbildung 4.2 skizziert, wird auch dabei ein zentraler Server verwendet, der die Audiodatenströme aller Teilnehmer empfängt und ein gemeinsames Signal generiert und allen zurücksendet. Das Erzeugen dieses gemeinsamen Signals ist grundlegend anders als beim NMP, denn hier gilt es nicht ein naturgetreues Mischsignal zu erzeugen, sondern viel mehr den gerade sprechenden Teilnehmer zu identifizieren und seinen Audiodatenstrom an alle zu verteilen. Funktional fällt dem Server so die Rolle des Moderators zu, der die Zuteilung des Rederechtes anhand der Audiodaten ermitteln muss. Üblicherweise wird das lauteste Signal ermittelt und der entsprechende Teilnehmer als aktiver Redner bestimmt.

Technisch gesehen enthält der Server einer Audiokonferenz keine Mischfunktionalität, es findet lediglich ein Multiplexen der Audiodatenströme und anschließender Punkt-zu-Mehrpunkt Reflexion statt. Das Fehlen strikter Vorgaben an Mischauthentizität und Synchronisationsgenauigkeit spiegelt die Grundverschiedenheit von NMP und Audiokonferenzsystemen wieder. Die Herausforderungen bei Audiokonferenzanwendungen sind komplett anders gelagert und gestalten sich aufgrund der viel höheren Latenztoleranz im Vergleich zu NMP als technisch deutlich weniger anspruchsvoll.

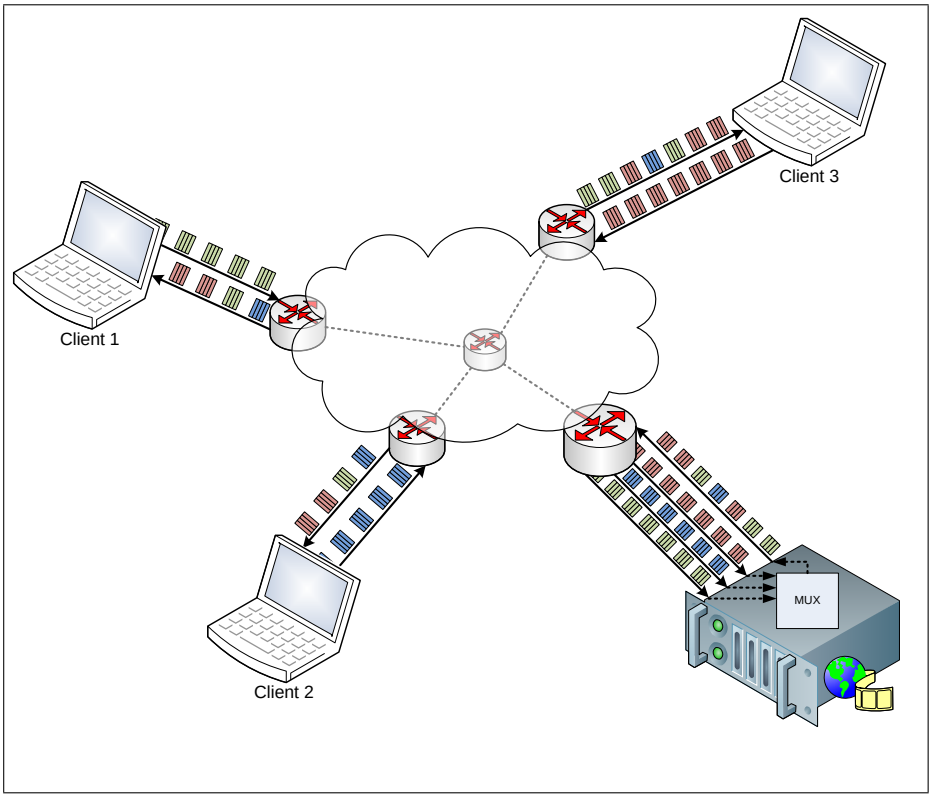

Abbildung 4.2: Prinzip einer Internet-Audiokonferenz

4.1.3 Netzwerkspiele

Der kontinuierliche Ausbau von kostengünstigen und breitbandigen Internetzugängen hat zu einem sprunghaften Anstieg bei der Nutzung von Online- bzw. Netzwerkspielen geführt. Nicht zuletzt durch die zusätzliche Attraktivität des netzverteilten Spielens gegen menschliche Gegner hat die Spieleindustrie mittlerweile als wirtschaftlicher Faktor die Filmindustrie überholt.

Es gibt unterschiedliche Klassen von Netzwerkspielen, die abhängig vom Interaktionsgrad sehr hohe Anforderungen an das Netz haben. Besonders die immer populärer werdenden Spielkonsolen fokussieren stark Action-lastige Spielinhalte, die eine hohe Interaktivität und geringe Latenzen erfordern. Bei höchst reaktionsschnellen Spielen wie 1st-Person-Shooter oder Autorennen liegt die Latenztoleranz in einem Bereich zwischen 50 und 100 ms. Damit ist, wenn wir die minimale Verzögerung als Ziel betrachten, NMP eher mit Netzwerkspielen als mit Audiokonferenzen vergleichbar. Diese Eigenschaft stellt sich jedoch auch als einzige Gemeinsamkeit heraus, die Herausforderungen bei Online-Spielen liegen in anderen Bereichen.

In Abbildung 4.3 ist der Aufbau eines Netzwerkspiels skizziert. Gemeinsam mit NMP ist dabei die Existenz eines zentralen Servers, an dem alle Spieler angebunden sind. Auf allen Clientrechnern und auf dem Server läuft eine Instanz des Spieles, in das die Spieler virtuell eintauchen. Dem Server fällt dabei die Aufgabe zu, die virtuellen Welten zu synchronisieren und so zusammenzuführen.

Das grundlegende Prinzip dabei ist für alle Online-Spiele identisch: die Clients senden dem Server jeweils in periodischen Abständen die vom Spieler getätigten Eingaben. Am Server werden alle in einem definierten Zeitabschnitt eintreffenden Ereignisse als gleichzeitige angenommen und in das laufende Spiel eingegeben. Die Auswirkungen der Eingaben auf den Spielverlauf werden berechnet und der aktuelle Zustand der virtuellen Spielewelt den Clients zurückgesendet.

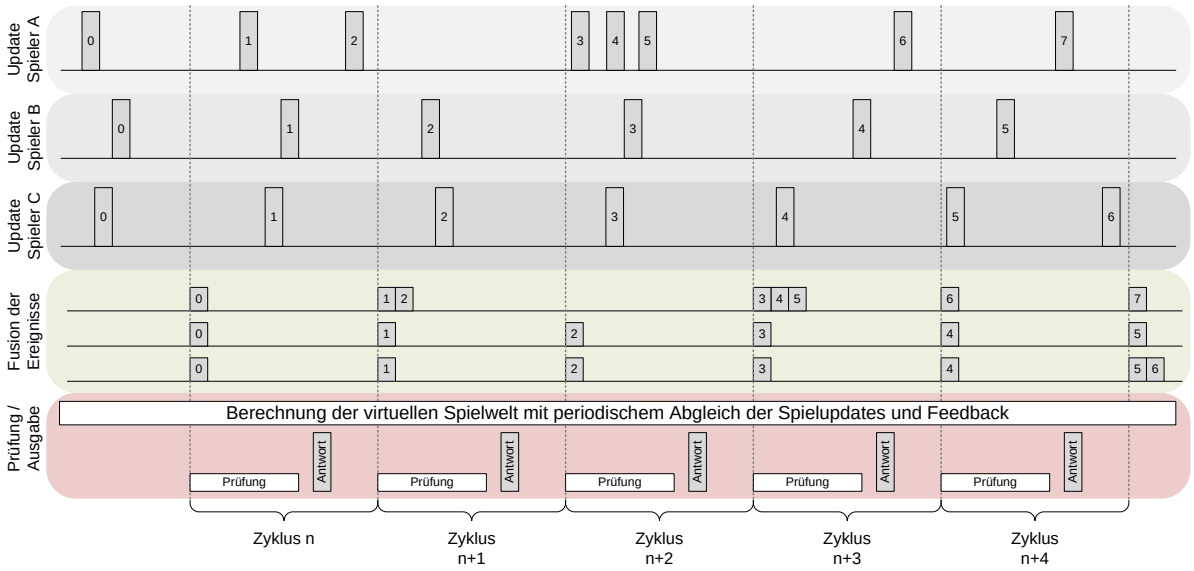

Abbildung 4.3: Ablauf eines Netzwerkspiels beim Server

Hoch interaktive Spiele können nur dann sinnvoll gespielt werden, wenn die Systemantwort auf Eingaben sehr gering ist. Zur Veranschaulichung: in einem Shooter werden Schüsse in Abständen von weniger als Zehntelsekunden abgefeuert und zwar in Abhängigkeit davon, ob das Ziel getroffen wurde. Innerhalb dieser Zeit muss die Eingabe des Benutzers zum Server versendet, das Ergebnis berechnet werden und wieder am Client angekommen sein. In dieser Hinsicht gleichen die Netzwerkspiele NMP am ehesten, da die Latenztoleranz von 50 ms nur wenig über der von NMP liegt, aber deutlich unter der für die betrachteten Audiokommunikationsanwendungen.

Um die erwarteten Antwortzeiten in solchen Echtzeitspielen einhalten zu können, müssen Aktualisierungsintervalle bis hinunter zu 10 ms eingehalten werden, d.h., es werden pro Sekunde 100 Pakete zwischen den Clients und dem Server ausgetauscht. Der Paketabstand ist so vergleichbar zu den Sprachanwendungen, die abhängig von verwendeten Audiocodec mit Blocklängen von 10 bis 20 ms arbeiten.

Anders als bei NMP oder den bisher betrachteten Audioanwendungen wird bei Netzwerkspielen kein kontinuierlicher Datenstrom übertragen. Dadurch sind die übertragenden Pakete relativ klein. Die resultierende Datenrate im Bereich zwischen zehn und 100 kbps liegt deshalb deutlich unter den Werten für NMP und macht den Zugang zu Netzwerkspielen auch bei schmalbandigen Anbindungen möglich.

Bedingt durch den nicht kontinuierlichen Charakter der übertragenen Daten entfällt bei Netzwerkspielen die Problematik der Synchronität: da es keinen lückenlosen Datenstrom gibt und die Daten nur periodisch ausgewertet werden, sind Maßnahmen zur Kompensation unterschiedlicher Paket- und Datenraten, wie sie in Audioanwendungen auftreten und in Abschnitt 6.7 diskutiert werden, nicht erforderlich.

4.1.4 Fazit

Diese kurze Auflistung von Anwendungen mit partiell ähnlich hohen Herausforderungen ist weder vollständig noch repräsentativ. Vielmehr zeigt sie auf, dass allgemein vertraute Anwendungen in isolierten Teilen ähnliche Anforderungen wie bei NMP zu erfüllen haben – das gemeinsame interaktive Musizieren erfordert dabei die Einhaltung aller Aspekte, die darüber hinaus jeweils die höchsten Anforderungen stellen.

Die relevanten Punkte sind in Tabelle 4.1 aufgeführt. Die im direkten Vergleich mit den anderen Audioanwendungen Internet-Telefonie und -Konferenz um eine Größenordnung geringere Latenztoleranz ist

Eigenschaft	VoIP	Audiokonferenz	Netzwerkspiele	NMP
Latenztoleranz [ms]	150-250	300-500	20-100	30
Paketabstand [ms]	10-20	10-20	10-100	2.66
Datenrate [kbps]	10-150	50-150	10-100	300-4608
Kollaboration	✗	✓	✓	✓
Mischen	✗	✗/✓ (Multiplexing)	✓	✓
Synchronisierung	✗	✗	✗/✓ (keine Kontinuität)	✓

Tabelle 4.1: Gegenüberstellung von NMP zu artverwandten Anwendungen

ursächlich für die ebenso deutlichen Unterschiede beim Paketabstand und bei der Datenrate verantwortlich. Die hohe Paketrate verursacht einen hohen Overhead, der, zusammen mit der Einschränkung, dass innerhalb der Latenztoleranz nur Audiocodierverfahren mit geringer Datenkompression verwendbar sind, zu einer unter den verglichenen Anwendungen deutlich höheren Datenrate führt.

Neben den Betriebsparametern unterscheiden sich die von den Anwendungen bereitgestellten Funktionalitäten deutlich: eine kollaborative Umgebung bietet die Audiokonferenz, Netzwerkspiele und NMP, während VoIP eher als ein bidirektionales Streaming zu betrachten ist. Ein Fusionieren bzw. Mischen der Daten findet beim VoIP prinzipbedingt nicht statt. Bei der IP-Audiokonferenz findet meist ein Quasi-Mischen in Form des Multiplexings statt. In Netzwerkspielen ist das Zusammenführen der Spielereignisse und der kontinuierliche Abgleich untereinander zentrale Funktion. Da diese jedoch im Vergleich zu NMP nicht kontinuierlich, sondern in äquidistanten und vom Server vorgegebenen Zeitabständen erfolgt, entfällt die Synchronisationsproblematik. Diese Herausforderung, kontinuierliche Datenströme unterschiedlicher Abtastraten zusammenzuführen, gilt unter den hier betrachteten Anwendungen nur für NMP.

Abseits der hier verglichenen Anwendungen scheinen andere vermeintlich höhere Anforderungen an die hier betrachteten Betriebsparameter zu stellen: intuitiv denkt man an Videokonferenzen, Tele-Operationen oder Telepräsenzanwendungen in virtuellen Räumen, die allesamt multimedialer und damit datenintensiver als NMP sind. Hierbei ist die Latenztoleranz vergleichbar hoch bzw. der Interaktionsgrad gering. Bei genauer Betrachtung findet bspw. bei der Tele-Operation gar keine Interaktion statt, da der Chirurg nach jeder Aktion zunächst beliebig lang auf das haptische oder visuelle Feedback warten und mit seiner weiteren Aktion abstimmen kann. Zwar erleichtert eine geringe Systemlatenz den Umgang mit einer solchen Anwendung, sie ist jedoch kein hartes Kriterium für deren generellen Einsatz.

Unter den heute im Alltag gebräuchlichen verteilten Netzanwendungen sind in lokalen Netzen gespielte Ego-Shooter diejenigen mit der geringsten Latenztoleranz. Mit NMP müssen wir ähnliche Werte erreichen, dabei jedoch kontinuierliche und lückenlose Datenströme mit deutlich höheren Daten- und Paketraten transportieren und synchronisieren.

4.2 Netzgestütztes Musizieren

Das IBR ist nicht Urheber der Idee, das Netz für gemeinsames verteiltes Musizieren einzusetzen. Vielmehr hat es entsprechende Versuche praktisch seit Anbeginn des Internets gegeben, das neue Medium als virtuelle Bühne für interaktive musikalische Zusammenkünfte zu nutzen. Die Ansätze haben sich dabei an dem jeweiligen Leistungsstand des Netzes orientiert, angefangen vom Austausch schmalbandiger Daten über musikalische Kompositionen bis hin zu heute machbaren immersiven Räumen, in denen die Musiker bzw. Zuhörer in virtuellen Orchestern vollständig eintauchen können.

In diesem Abschnitt wollen wir einen kurzen Abriss über relevante Ansätze des netzgestützten Musizierens präsentieren. Wir beschränken uns dabei auf diejenigen Arbeiten, die uns zur Bestandsaufnahme am Projektstart etwa im Jahre 2006 zugänglich waren. Auf aktuellere Entwicklungen, die im Laufe der Arbeiten an NMP erfolgt sind, gehen wir am Schluss der Arbeit in Kapitel 10 ein.

4.2.1 Ansätze mit synthetischen Audiodaten

In der Anfangszeit des Internets waren die verfügbaren Kapazitäten des Netzes so eingeschränkt, dass an einer Echtzeit-Übertragung natürlicher Audiodaten nicht zu denken war. Stattdessen wurde zunächst übergangsweise an Verfahren gearbeitet, die auf den Austausch synthetischer Daten beruhen. Diese beschreibende Form musikalischer Daten beruht darauf, dass statt abgetasteter Audiosamples lediglich die Beschreibung der Klangerzeugung und ihrer zeitlichen Abfolge festgehalten wird. Unter der Prämisse, dass bspw. ein Klavier beim Betätigen einer Taste mit bestimmten Aktionsparametern zu einer deterministischen Klangerzeugung führt, nutzen solche Formen musikalischer Datenrepräsentation dieses Wissen bei jedem Teilnehmer. Sie fassen ein Musikstück als Überlagerung zeitlicher Abfolge von Aktionsereignissen für definierte Musikinstrumente auf. Jeder Teilnehmer verfügt implizit über die Informationen, wie eine Aktion zu dem zugehörigen Audiosignal umzusetzen ist – sei es, dass sie direkt über betreffenden Instrumente ausgeführt oder durch simulierte Klangerzeuger in Wave-Tables (siehe [37]) am PC generiert werden.

Die am meisten verbreitete und noch heute verwendete beschreibende Sprache in der Musik ist MIDI (Musical Instrument Digital Interface [64]), in dem die Übermittlung musikalischer Steuerinformationen definiert ist und heute praktisch von allen Herstellern digitaler Musikinstrumente und -Zubehör unterstützt wird.

Die Vorteile bei der Verwendung von MIDI für das verteilte Musizieren im Netz liegen auf der Hand: die für die Beschreibung der Steuerinformationen erforderliche Datenmenge ist im Vergleich zu digitalem Audio ungleich kleiner. Während wir für NMP bei einer Samplingrate von 48 kHz bei 16 bpS mit einer kontinuierlichen Datenrate von 768 kbps arbeiten, fallen bei MIDI nur dann Daten an, wenn Aktionsänderungen auftreten. Da diese nicht kontinuierlich und darüber nur einen geringen Informationsgehalt haben, fällt die zu bewältigende Datenrate so gering aus, dass für deren Übertragung bereits Analogmodems ausreichend sind und somit die Machbarkeit netzverteilten Musizierens bestand, lange bevor MP3 die Verteilung digitaler Musik eröffnete.

Als ein relevanter Vertreter dieses Ansatzes soll hier das Virja Projekt in [75] genannt werden, mit dem die Realisierbarkeit verteilter Jazz-Sessions im Netz untersucht wurde. Auf kommerzieller Ebene war zu dieser Zeit das Unternehmen eJamming (`http://www.ejamming.com`) aktiv, welches seinen Kunden eine kostenpflichtige Lösung zum MIDI-basierten Musizieren im Internet anbot.

Die Vorteile von MIDI hinsichtlich seines geringen Datenaufkommens implizieren gleichzeitig seine Nachteile, sobald es um die Detailtreue der abzubildenden Musik geht. Zunächst schränkt die bereits im Akronym enthaltene Limitierung auf digitale Geräte einen Großteil von Instrumenten aus. Praktisch anwendbar ist es für solche, bei denen sich die Klangerzeugung mit individuellen und wohl definierten diskreten Parametern beschreiben lässt. Gut geeignet sind dabei Tasten, Zupf- und Schlaginstrumente, da sie durch definierte Parameter (wie Geschwindigkeit und Stärke des Anschlages) aktiviert und deaktiviert werden. Dementsprechend werden alle Instrumente nicht unterstützt, die eine kontinuierliche Interaktion mit fortwährender Einwirkung erfordern. Streich- und Blasinstrumente entfallen daher und schließen folglich klassische Musik aus dem Anwendungsspektrum aus. Auch im optimalen Fall kann MIDI keinen natürlichen Musikeindruck vermitteln, da es nicht alle Parameter für die Klangerzeugung abdeckt und die Nuancen, die Musik lebendig machen, nicht abbilden kann – die generierte Musik klingt synthetisch.

Für den praktischen Einsatz von NMP noch gravierender ist die fehlende Unterstützung für Gesang und Sprache, da eine verbale Interaktion fehlt, die das entscheidende Werkzeug für das Aufsetzen, Durchführen und Bewerten von Sitzungen darstellt.

Technisch betrachtet stellt das netzgestützte Musizieren mit MIDI-Daten komplett andere Anforderungen als das mit natürlichem Audio. Das Fehlen von Kontinuität und der ereignisbasierte Charakter bewirken, dass ein MIDI-System sich eher mit den Problematiken von Netz-Spielen befassen muss, die wir weiter oben bereits aufgeführt haben. Das betrifft insbesondere die Behandlung von Paketverlusten: wenn bei der Übertragung ein MIDI-Event verloren geht, führt es dazu, dass entweder das Anschlagen (*Note-On*) einer Taste oder das Lösen (*Note-Off*) verloren geht. Im ersten Fall führt es dazu, dass eine Note vollständig fehlt, im zweiten wird ein aktivierter Ton fälschlicherweise dauerhaft gehalten. Die Herausforderungen bei der Bewältigung von Paketverlusten bestehen daher darin, kontinuierlich Plausibilitätsprüfungen durchzuführen und potenzielle Datenverluste zu korrigieren.

Alle Probleme in NMP, die aufgrund des kontinuierlichen Charakters der ausgetauschten Daten anfallen, kommen bei einem MIDI basierten System nicht zum Tragen und verhindern damit eine Vergleichbarkeit.

4.2.2 Quasi-verteilte Szenarien zwischen Bühne und Auditorium

Manche existierende Arbeiten deklarieren Szenarien als netzverteilt, welche technisch die wesentlichen für NMP angenommenen Kriterien nicht erfüllen. Dabei geht es um Projekte, die den Aspekt der verteilten Präsenz auf das Erweitern einer lokal gehaltenen musikalischen Sitzung auf zusätzliche passive Teilnehmer beschränken. Man denke dabei an eine Orchesteraufführung, die über das Netz an verschiedene Orte übertragen wird, an denen der Eindruck einer Liveveranstaltung entsteht.

Eine verteilte Interaktivität gibt es in solchen Szenarien nur sehr eingeschränkt oder gar nicht: die Musiker führen ihr Stück wie gewohnt auf und interagieren mit dem Publikum allenthalben dahin gehend, dass deren Reaktion in die Rückrichtung übertragen und der Beifall vernommen wird. Technisch sind solche Anwendungen ohne große Anstrengungen umsetzbar, da sie lediglich die aus etablierten TV-Liveübertragung bekannten Mechanismen auf das Medium Internet verlagern. Die für die Durchführung erlaubten Verzögerungszeiten von mehreren Sekunden ermöglichen einen Betrieb über gewöhnliche verbindungsorientierte Kanäle und ohne die bei NMP üblichen Schwierigkeiten.

Als frühe Vertreter solcher Ansätze lassen sich Projekte wie das in [98] beschriebene anführen. Entsprechende Funktionalität ist mittlerweile in professionellen und semiprofessionellen Musikanwendungen eingeflossen, die den zeitnahen Austausch von Audiospuren über das Streaming in virtuellen Studios ermöglichen, wie bspw. DML (Digital Musician Link, `http://www.digitalmusician.net`) von Steinberg.

Ein leicht erhöhter Interaktivitätsgrad wird erforderlich, wenn in Lehr-Lernszenarien die Zuhörer nicht vollständig passiv sind. Solche Lehrer-Schüler Aufbauten werden in der Literatur bspw. in [54, 55] aufgegriffen. Gegenüber der reinen Liveübertragung erfordert eine praktikable Interaktion des Lehrers mit seiner Klasse geringere Antwortzeiten.

Die in der Praxis zu erfüllenden Anforderungen konnten wir während der Mitarbeit an einem von der *ECMA* (European Chamber Music Academy) geförderten Projekt für den netzbasierten Unterricht untersuchen [50]. Ziel dieses Projektes war eine Prüfung, inwieweit die heute verfügbaren Möglichkeiten der Netztechnologie im Lehralltag erforderlichen Reisetätigkeiten von Musikstudenten und Professoren reduzieren könnten.

In diesem speziellen Fall sollten die Musikstudenten wie gehabt an einem Ort üben. Der Professor sollte über eine audiovisuelle Anbindung an den Proben virtuell teilhaben, Verbesserungen und Kommentare einbringen und die Leistungen der Studenten beurteilen können. Im Verlauf des Projektes hat sich heraus-

gestellt, dass sich die Anforderungen dabei auf eine möglichst naturgetreue Abbildung der Sound-Kulisse fokussierten. Der Bedarf nach einem hohen Interaktionsgrad hat sich dagegen nicht bestätigt. Der Professor greift üblicherweise erst in das Geschehen hinein, wenn eine Passage vollständig gespielt wurde. Seltener erfolgten Korrekturen oder Anweisungen mitten im Spiel, auch dabei wurde eine Latenz von bis zu einer Sekunde als tolerierbar bewertet.

Angesichts dieser begrenzten Anforderungen wurde das Projekt schließlich auf Basis bereits existierender Videokonferenzsysteme aufgebaut, die eine ausreichend hohe Audioqualität bereitstellen. Die etablierten Verfahren zur Datenübertragung und Bearbeitung von Paketverlusten können den erforderlichen Anforderungen für solche Szenarien genügen.

4.2.3 Quasi-gleichzeitiges Musizieren

Eine Gruppe von Projekten betrachtet die NMP-Problematik angesichts der im Internet schwer zu gewährleistenden Latenzen weniger strikt und verfolgt den Ansatz der Takt-Synchronität. Dabei wird der für die direkte Improvisation erforderliche Interaktionsgrad bewusst zurückgefahren zugunsten eines Schemas, das die Einhaltung eines einheitlichen Taktes bei allen Teilnehmern sicher stellt.

Als Repräsentanten dieser Klasse von Anwendungen greifen wir die frühen Arbeiten in [14] auf, die im weiteren Verlauf mit der Software *NinJam* [13] realisiert wurde. Die Grundidee bei diesem Ansatz ist die Verwendung einer zentralen Komponente, die zur Synchronisation einen gemeinsamen virtuellen Taktgeber bzw. Dirigenten allen Teilnehmern bereitstellt. Die Musiker treten nach und nach der Sitzung bei und spielen das Instrument bzw. den Gesang unter Beachtung des gegebenen Taktes ein. Abhängig von der Übertragungsverzögerung ihrer Daten auf dem Weg zum Server wird jeder Beitrag einem bestimmten Takt zugeordnet. So erhalten diejenigen Teilnehmer, deren Nähe zum Server sich in kurzen RTT Werten niederschlägt, die Reaktion mit einer kurzen Antwortzeit, während solche mit langer RTT auch mal mehrere Takte auf das Feedback ihrer Eingabe warten müssen.

Dieser Ansatz zieht die Konsequenz daraus, dass gemeinsames Musizieren, so wie wir es in unserer Arbeit anstreben, nur innerhalb eines überschaubaren Anwendungsradius machbar ist und versucht, diese Limitierung durch die Erhöhung der Toleranzschwelle auf Vielfache der Taktzeit auszudehnen. Eine unmittelbare Interaktion der Musiker untereinander, d.h., die direkte Antwort auf eine improvisierte Änderung, ist nicht möglich, da diese frühestens im folgenden Takt wahrnehmbar ist. Die Autoren des Verfahrens heben hervor, dass die Benutzung von *NinJam* eine gewisse Einarbeitung erfordert. Angesichts des geringeren Interaktivitätsgrades ist davon auszugehen, dass sich die Anwendung darüber hinaus für bereits einstudierte Stücke ohne signifikante Abweichungen eher eignet, als für solche, die erst beim Spielen entstehen.

Mit NMP verfolgen wir einen *NinJam* entgegengesetzten Ansatz, indem wir einen reduzierten Aktionsradius in Kauf nehmen, dafür jedoch eine minimale Systemlatenz anstreben, die jegliche Art des gemeinsamen Musizierens erlaubt.

4.2.4 Anwendungen in geschlossenen Netzen

Angesichts der beschriebenen Herausforderungen und den im Internet vorherrschenden Bedingungen für die Umsetzung Netz verteilten Musizierens schränken manche Ansätze ihren Anwendungsbereich auf bestimmte Netze ein. Als Vertreter dieser Anwendungsklasse führen wir hier die Ansätze auf, die innerhalb des Internet2 erforscht werden.

Zu den ersten Projekten, die dieses hochleistungsfähige Forschungsnetz für die Übertragung interaktiver Musikdaten einsetzen, können die am CCRMA Institut der Stanford Universität innerhalb der SoundWire

Arbeitsgruppe durchgeführten Arbeiten genannt werden. In [20] werden die ersten Ansätze beschrieben, vorher im LAN erfolgte Untersuchungen auf das Internet2 auszudehnen.

Das Streamen von Live-Sitzungen zur Realisierung von Tele-Präsenzen wird an der McGill Universität am Institut CIRMMT (Centre for Interdisciplinary Research in Music Media and Technology) erforscht. Innerhalb der heutigen Forschungsgruppe Ultra-Videoconferencing wird das Thema in einer großen Bandbreite bearbeitet, angefangen von der Verwaltung und Übertragung der bei Konzerten anfallenden enormen Datenmenge bis hin zur Realisation virtueller Bühnen mit hoher Immersion. In [22] beschreibt die Arbeitsgruppe, wie erstmalig über das Forschungsnetz eine Live-Sitzung über das Internet2 über große Distanzen gestreamt wird. Basierend auf dieser Machbarkeitsstudie wurden in den folgenden Jahren verschiedene Formen des interaktiven Musizierens untersucht und demonstriert, die im Wesentlichen die von unserem NMP anvisierte Funktionalität vorwegnehmen.

Allerdings sind die Zielsetzungen dieser Arbeiten nicht mit unseren vergleichbar, was an den zu bewältigenden Herausforderungen deutlich wird. So zielen die auf das Internet2 aufsetzenden Arbeiten meist darauf ab, die Möglichkeiten und Leistungsfähigkeit des neuen Netzes zu demonstrieren, bspw. dadurch, dass eine Vielzahl paralleler Audio- und Videodatenströme unkomprimiert übertragen wird. Während dabei also der Bedarf für das künftige Netz untermauert wird, wollen wir mit NMP die Anwendungsgrenze des heutigen Internets auf das interaktive Musizieren ausdehnen.

Entsprechend unterscheiden sich die Herausforderungen bei der Umsetzung der Vorhaben. Als wesentlichen vereinfachenden Faktor ist hierbei die Verfügbarkeit von QoS im Internet2 aufzuführen [91], die praktisch alle bei NMP im Internet zu erwartenden Schwierigkeiten löst. Mit dem Einsatz der Dienstgütemechanismen entfällt der in Abschnitt 2.2.1.5 beschriebene stochastische Charakter der Übertragungszeit und erübrigt jegliche De-Jitter Maßnahmen, die wir in NMP vornehmen müssen.

4.2.5 Dezentrale Verfahren

In verschiedenen Anwendungen ist die Audiodatenübertragung als isolierte Komponente heute bereits etabliert. Das unter Linux für zeitkritische Audioverarbeitung eingesetzte Jack Audio-Subsystem [24] stellt Client- und Server-Module bereit, mit denen die Audiodaten zwischen Rechnern innerhalb eines lokalen Netzes ausgetauscht werden können. Neben einer rudimentären Behandlung von Paketverlusten bieten bestimmte Implementierungen der Module Unterstützung für das Angleichen von Abtastratenabweichungen.

Die Beschränkung der uni-direktionalen Datenübertragung vom Server zum Client bei Jack begegnet LDAS (Low Delay Audio Streamer [6]), ein Werkzeug für den bidirektionalen Austausch von Audiodaten zwischen zwei Rechnern im Netz. Es wurde im Rahmen von Forschungsarbeiten für die Verwendung unter Linux entwickelt und stellt die Funktionalität einer Netz-Audiokarte bereit. Gezielt für den Einsatz in Szenarien mit geringer Latenz entwickelt, werden für NMP relevante Aspekte wie Synchronisation, Abtastratenanpassung und Abspielverzögerung behandelt.

So decken Jack und LDAS die für die Datenübertragung bestimmten Herausforderungen ab. Sie beschränken sich jedoch auf eben diese zentrale Komponente und lassen sich nicht explizit für das interaktive Musizieren mit mehreren Teilnehmern verwenden.

Einen Ansatz, diese elementaren Funktionen in ein System zum netzverteilten Musizieren auszubauen verfolgt die Arbeit in [18]. Das innerhalb des dortigen – ebenfalls NMP genannten – Projekt entwickelte Werkzeug *SoundJack* unterstützt die zeitnahe Übertragung von Audiodaten zwischen Teilnehmern und ermöglicht das interaktive Musizieren im Netz. Im Unterschied zu unserem zentralistischen System verwendet *SoundJack* eine Multi-Punkt-zu-Punkt Architektur, in der jeder Client seine Daten jedem anderen zusendet und von diesem jeweils einen Audiodatenstrom empfängt.

Durch den gemeinsam verfolgten Ansatz, die Systemlatenz innerhalb der Toleranzgrenzen zu halten, ist dieses System unserem NMP allen uns bislang bekannten am ähnlichsten. Das Fehlen eines zentralen Servers innerhalb des gewählten P2P Designs macht die Ansätze jedoch nicht vergleichbar.

Wir haben bereits in Abschnitt 2.3 unsere Wahl der Systemarchitektur begründet und dabei herausgestellt, dass eine Funktionalität, die wir für unser NMP erreichen wollen, ohne zentrale Synchronisations- und Datenaggregationseinheit nicht realisierbar ist. Dementsprechend fehlt dem P2P-NMP jegliche nachgelagerte auf Datenaufzeichnung beruhende Funktionalität, mit der unser System virtuelle Clients und Bands umsetzt. Viele der in Kapitel 3 genannten Herausforderungen treffen für eine dezentrale Anwendung nicht zu und erübrigen somit einen Großteil der in Kapitel 6 diskutierten Lösungsansätze.

4.2.6 Fazit

Der Umfang verwandter Arbeiten zum Thema interaktives Musizieren im Netz ist breit gefächert: angefangen von auf MIDI basierenden Ansätzen, die das Zusammenspiel auch in schmalbandigen Netzen erlauben, über Projekte, die sich einzelner Aspekte der Echtzeit-Audioübertragung annehmen, bis hin zu Systemen, die virtuelle audiovisuelle Bühnen mit hohem Immersionsgraden schaffen, ist alles vertreten.

Von den hier vorgestellten und in Tabelle 4.2 zusammenfassend aufgeführten Arbeiten ist keine mit unserem NMP-System direkt vergleichbar. Diejenigen, die das heutige Internet als Medium nutzen, sind angesichts der bekannten Einschränkungen und Herausforderungen in ihrer Funktionalität eingeschränkt oder versuchen erst gar nicht, die für transparentes Musizieren erfolgreiche Latenzschwelle einzuhalten. Diejenigen, die auf künftige Netze wie dem Internet2 aufsetzten, sind aufgrund der verfügbaren Dienstgütemechanismen mit keiner Variation der Übertragungsverzögerung konfrontiert. Ohne den Jitter entfällt der Ursprung vieler Herausforderungen unseres NMP-Systems und der erforderlichen Kompensationsmaßnahmen. Angesichts der verfügbaren Netzleistung zielen die Projekte im Internet2 eher darauf ab, mit neuen Anwendungsformen den Bedarf für solche Netze zu rechtfertigen.

Ein neben den funktionalen Unterschieden in allen betrachteten Arbeiten fehlender Aspekt ist ein ganzheitlicher Systemgedanke, den wir bei der Umsetzung von NMP verfolgt haben. Dabei haben wir aus der empirisch ermittelten Latenztoleranz und einer analytisch ermittelten unteren Latenzschranke ein Latenzbudget aufgestellt, dem wir jegliche Realisierungsmöglichkeit untergeordnet haben. Verfahren oder Prozesse, die das Budget überschreiten (bspw. effiziente Audiokompressionsverfahren mit höheren algorithmischen Verzögerungen), haben wir bewusst ausgeschlossen, um die Latenztoleranz nicht zu überschreiten.

Zwar sind die entsprechenden Herausforderungen auch allen aufgeführten Projekten bekannt, diese werden an vielen Stellen jedoch als temporär und als Teil der technischen Entwicklung von sich selbst ver-

Latenztoleranz $[ms]$	Datenrate $[kbps]$	Echtzeit-Interaktion	Zentrales Mischen und Synchronisation	Ohne QoS
Synthetisches Audio: Virja, eJamming				
30-50	2-10	✓	✗	✓
Quasi-verteilt: Lehrer-Schüler (ECMA), Musiker-Publikum, Musiker-Musiker (DML)				
500-3.000	800-10.000	✗	✗	✓
Quasi-gleichzeitig: NinJam				
200-5.000	100-600	✗	✓	✓
Geschlossene Netze: SoundWire, Ultra-Videoconferencing				
50-250	1.500-2.000.000	✓	✓	✗
Dezentral: Jack, LDAS, SoundJack				
10-150	100-2.000	✓/✗	✗	✓

Tabelle 4.2: Verwandte Arbeiten zum netzverteilten Musizieren

schwindend eingestuft. Wir werden in dieser Arbeit zeigen, dass diese Annahme nicht zutrifft und interaktives netzverteiltes Musizieren immanenten Einschränkungen unterliegt.

Wir sehen daher unsere Ergebnisse als Teil eines Fundamentes an, welches sowohl für die existierenden Arbeiten gültig ist als auch für künftige Ansätze eine nicht überbrückbare untere Schranke bei der Realisierung darstellt.

KAPITEL 5

LATENZANALYSE

Unter allen in Kapitel 3 dargestellten Herausforderungen nimmt die Latenz in einem NMP-System eine besondere Stellung ein: als einziges Kriterium ist sie obligatorisch und bestimmt die Realisierbarkeit netzgestützten Musizierens. Alle beschriebenen Anforderungen zur Audioqualität, Synchronität und Stabilität verbessern die Akzeptanz von NMP, die durch die Einhaltung einer Verzögerung unterhalb der Wahrnehmungsschwelle erst ermöglicht wird.

Diesem fundamentalen Kriterium wenden wir uns in diesem Abschnitt zu. Wir betrachten zunächst, an welchen Stellen und warum Verzögerungen innerhalb eines NMP-Systems auftreten. Nach der Klassifizierung werfen wir einen genaueren Blick auf die Einzelkomponenten und bestimmen die auftretenden Latenzen in den Endsystemen (Client und Server) und im Netz. Anschließend quantifizieren wir die auf den Datenpfad innerhalb der Endsysteme existierenden und unvermeidbaren Verzögerungen und bestimmen damit die untere Schranke der Systemlatenz in Client und Server. Der ermittelte Wert legt zusammen mit der Latenztoleranz das Budget an Verzögerung fest, welche für die Datenübertragung im Netz verfügbar ist. Mit dieser Größe schätzen wir die obere Schranke für den Einsatzradius eines NMP-Systems ab und betrachten abschließend seine Gültigkeit in realer Umgebung.

Die Verifizierung der durchgeführten Analyse erfolgt mit einem latenzoptimierten NMP-Minimalsystem, welches alle übrigen Herausforderungen ignoriert und ausschließlich darauf abzielt, die theoretisch bestimmten Schranken zu realisieren. Wir beschreiben dabei, wie ein solches System zu entwerfen ist, und diskutieren die kritischen Komponenten. Mit unseren Messergebnissen bestätigen wir abschließend, dass die analytisch ermittelten Werte für die Systemlatenz in den Endsystemen erreicht werden und bei Einhaltung vorgegebener Netzeigenschaften ein Anwendungsradius für NMP abgeschätzt werden kann, innerhalb dessen netzverteiltes Musizieren praktikabel ist.

5.1 Ursachen für Verzögerungen in einem NMP-System

In unserem NMP-System lassen sich die auf den Datenpfad auftretenden Verzögerungen zunächst grob wie in Abbildung 5.1 dargestellt in drei Abschnitte aufteilen. In den Endsystemen werden grundlegend unterschiedliche Verarbeitungen durchgeführt, weswegen zwischen der Verweildauer der Daten im Client (t_{CP}) und im Server (t_{SP}) unterschieden wird, während die im Netz auftretenden Verzögerungen (t_N) gänzlich an-

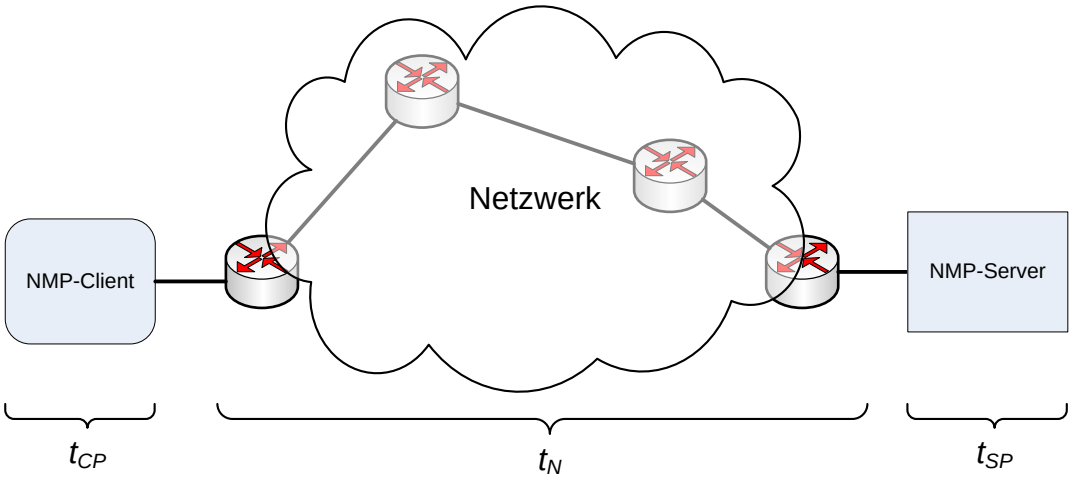

Abbildung 5.1: Latenzen auf dem Datenpfad in einem NMP-System

dere Charakteristiken haben. Unabhängig von der Verteilung der Gesamtlatenz gilt so formal $t = t_{CP} + t_{SP} + t_N$.

Für eine weitere Betrachtung der Verzögerung und ihrer Abschätzung ist eine Klassifizierung sinnvoll. Wir wollen anschließend zwischen deterministischen und statistischen Verzögerungen unterscheiden. Für die Bestimmung der unteren Schranke können wir anschließend die statistische Komponente vernachlässigen und uns auf die systemimmanenten deterministischen Anteile beschränken.

Zu den vorhersagbaren Verzögerungen gehören alle physikalischen oder technischen Gegebenheiten, die die Datenverarbeitung mit einer vorgegebenen Geschwindigkeit durchführen bzw. verzögern. Beispiele dafür sind:

Verzögerungen durch Pufferung

Datenverarbeitung läuft meist blockbasiert ab, Daten werden also nicht atomar und kontinuierlich bearbeitet, sondern erst gesammelt und dann im Block behandelt. Dieses trifft für die Verarbeitung in Hard- wie auch in Software zu und ist nicht selten der einfacheren Verarbeitung innerhalb eines Multiprozess-Betriebssystems geschuldet. Oft ist eine Pufferung jedoch auch prinzipbedingt nicht anders möglich: so kann beispielsweise auf eine Festplatte nur ein ganzer Block gelesen oder geschrieben werden bzw. ein Audiocodierer nur auf einen vollständigen Frame angewendet werden. In einer IP-basierten Netzanwendung ist eine blockweise Operation dadurch vorgegeben, dass die Kommunikation über Datenpakete erfolgt, die zuvor erst gepuffert werden. Liegen die zu bearbeitenden Daten in konstanter Datenrate vor, ist die Pufferlatenz abhängig von der Pufferlänge und der Datenrate und lässt sich nach Formel (5.1) berechnen. Verwenden zwei hintereinander geschaltete Funktionseinheiten unterschiedliche Puffergrößen kann nach Formel (5.2) eine Minimierung der kombinierten Latenz dann erreicht werden, wenn eine Anwendung durchgängig alle Puffer als Vielfache einer definierten atomaren Puffereinheit dimensionieren. Latenzoptimal arbeitet eine Anwendung dann, wenn alle verwendeten Puffer die gleiche Größe haben und damit ein Pipelining möglich ist.

Übertragungsverzögerung

Physikalische Gesetzmäßigkeiten wie die Signalausbreitungsgeschwindigkeit in Kupferleitungen oder die Schallausbreitung in Luft stellen die äußersten und unverrückbaren Schranken dar, während die technischen Grenzen meist deutlich das physikalische Limit noch nicht erreichen. Bei der Abschätzung der Verzögerung bei der Datenübertragung kann auf die nominelle Übertragungsgeschwindigkeit des verwendeten Mediums zurückgegriffen werden. Bei den heutigen Rechnerarchitekturen sind diese mit den vorhandenen Datenbussen innerhalb des Rechners praktisch vernachlässigbar, relevant werden

die Latenzen meist erst, wenn der Datenpfad den Rechner verlässt und über angeschlossene Peripherie weiter geleitet wird.

Verarbeitungszeit

Ähnlich wie die Datenübertragung erfolgt die Datenverarbeitung mit einer vorgegebenen Verarbeitungsgeschwindigkeit, die im Wesentlich durch die Eigenschaften der CPU und ihrer Taktung bestimmt wird. Eine Abschätzung der CPU-Zeit kann anhand der Taktzykluszeit und der für die Abarbeitung eines Algorithmus benötigten Anzahl von Instruktionen vorgenommen werden.

Statistische Latenzen beruhen auf adaptiven Verfahren, die ihre Abläufe dynamisch den Gegebenheiten zur Laufzeit anpassen und daher ein nicht vorhersagbares Verhalten haben. Dazu gehören:

Verzögerungen durch Ressourcenteilung

Arbeitsplatzbetriebssysteme sind heute durchweg multiprozess- und mutlithreadfähig, sie unterstützten das quasi-gleichzeitige Abarbeiten verschiedener Aufgaben. Dabei werden verfügbare Systemressourcen wie CPU, Speicher, Peripherie oder Datenbusse den konkurrierenden Prozessen nacheinander für eine vorgegebene Zeit zugewiesen und danach wieder entzogen. Diese Arbitrierung in Hardware bzw. Scheduling in Software verursacht eine variable Verzögerung. Der Wechsel zwischen den Prozessen findet so schnell statt, dass der Benutzer den Eindruck gewinnt, alle ausgeführten Programme liefen tatsächlich gleichzeitig ab. Die Zuteilung der Ressourcen wird im Betriebssystem vom Scheduler verwaltet, der diese nach unterschiedlichen Kriterien wie Fairness oder Priorisierung vornimmt. Dem Mehrprozessbetrieb wohnt daher eine Wartezeit auf die Zuteilung der Systemressourcen inne, die abhängig ist von der Auslastung des Systems und der gewählten Strategie des Schedulers. Ähnliche Verzögerungen bei Ressourcenkonflikten treten bei der paketvermittelten Datenübertragung in Netzen auf. Die Ressource kann dabei bspw. das Übertragungsmedium bzw. der Router sein.

Datenabhängigkeiten

Die Ausführungszeit von Algorithmen und Operationen ist nur dann vorherbestimmbar, wenn bei jedem Durchlauf identische Daten verwendet werden. Im praktischen Betrieb werden dagegen dynamische Daten verarbeitet, sodass sich die algorithmischen Abläufe unterscheiden und zu variierenden Bearbeitungszeiten führt.

Diese Unterteilung ist nicht vollständig oder allgemeingültig, gleichwohl für unsere Betrachtung gut geeignet. Sie hilft in der folgenden Analyse, Latenzen in den Endsystemen und im Netz nach ihrem Beitrag zur Gesamtverzögerung zu sortieren und unvermeidbare von vermeidbaren Anteilen zu unterscheiden. Es ist

Paketisierungslatenz: Wird ein Datenstrom mit der konstanten Datenrate von D_r in Paketen konstanter Größe B_l paketisiert, entsteht eine Paketisierungslatenz

$$t_B = \frac{B_l}{D_r} \tag{5.1}$$

Verwenden zwei hintereinander geschaltete Funktionseinheiten die gleichen Puffergrößen B_{l_1} und B_{l_2}, treten keine zusätzlichen Verzögerungen auf. Ist B_{l_2} größer als B_{l_1}, hängt das Pufferdelay davon ab, ob die beiden Werte teilerfremd sind. Ist das nicht der Fall, liegt die Gesamtverzögerung bei t_{B_2}, andernfalls beim kleinsten Vielfachen von t_{B_1}, welches t_{B_2} übersteigt. Es gilt:

$$B_{l_2} \geq B_{l_1}, \quad t_B = \begin{cases} t_{B_1} = t_{B_2} & \text{wenn } B_{l_1} = B_{l_2} \\ t_{B_2} & \text{wenn } B_{l_2} = n \cdot B_{l_1}, n \in \mathbb{N} \\ \left\{ \left\lceil \frac{B_{l_2}}{B_{l_1}} \right\rceil \cdot \frac{t_{B_1}}{D_r} \right\} & \text{sonst} \end{cases} \tag{5.2}$$

Beispiel: $D_r = 1000Hz, B_{l_1} = 4, B_{l_2} = 9 \quad \Rightarrow \quad t_B = \left\lceil \frac{9}{4} \right\rceil \cdot \frac{4}{1000Hz} = 12\,ms$

dabei zu beachten, dass die beiden Latenztypen selten alleine vorkommen. So setzt sich bspw. die Übertragungsdauer eines Datenblocks über einen Bus aus der berechenbaren Übertragungsverzögerung und der statistischen Wartezeit für die Arbitrierung des Busses zusammen. Das Vorgehen bei der Abschätzung der Latenz unterscheidet sich für beide Arten wesentlich: die deterministischen Verzögerungen können wir analytisch bestimmen und praktisch verifizieren, während wir für die statistischen Anteile empirische Abschätzungen vornehmen müssen.

Eine verlässliche Abschätzung der zu erwartenden Latenzen in den Endsystemen erfordert eine genauere Betrachtung der technischen Grundlagen von Rechnerarchitekturen und der Abläufe in heutigen Arbeitsplatzrechnern. Der folgende Exkurs soll einen kurzen Einblick in die für NMP relevanten Bereiche Rechnerhardware und Betriebssysteme vermitteln. Fachkundige können diesen Abschnitt überlesen und direkt zu Abschnitt 5.3 springen.

5.2 Exkurs: Technische Grundlagen von Arbeitsplatzrechnern

Unser Ziel beim NMP ist der Einsatz des Systems auf vorhandenen Rechnern, wie sie mittlerweile in nahezu allen Haushalten vorzufinden sind. Als Betriebssysteme wird man in den allermeisten Fällen auf Microsoft Windows, Linux oder MacOS treffen, allesamt Multiprozess-Betriebssysteme. Warum an gewissen Orten in den Endsystemen Verzögerungen auftreten und welche davon nicht zu vermeiden sind, erfordert ein Grundverständnis über Rechnerstrukturen und die Abläufe in Betriebssystemen. Aus Platzgründen können wir uns dabei nur auf die fundamentalen und für NMP relevanten Zusammenhänge beschränken, eine umfassende und weiterführende Behandlung der Thematik wird in der Literatur breit aufgegriffen, als ein Standardwerk empfiehlt sich bspw. [90]. Bei der folgenden Betrachtung teilen wir das System in seine drei Bestandteile User-Space, Kernel-Space und Hardware auf und betrachten die Latenzquellen innerhalb der beiden letztgenannten Komponenten.

5.2.1 Kernel-Space

Der Kernel stellt die zentrale Komponente eines Betriebssystems dar. Er verwaltet die Ressourcen des Rechners (Hardware, Speicher, CPU), regelt die Prozessabläufe und stellt die Schnittstelle zwischen User-Space und Hardware bereit.

5.2.1.1 Zugriff von Anwendungen auf Hardware über Gerätetreiber

Der Zugriff der Anwenderprogramme auf Peripherie erfolgt über Gerätetreiber, die im Kernel ausgeführt werden. In Ausgaberichtung stellt der Treiber der Anwendungsebene spezifizierte Schnittstellen zum Zugriff auf das Gerät bereit. In Eingaberichtung reagiert der Treiber auf Benachrichtigung der Hardware und leitet Maßnahmen zur Kontrolle des Gerätes oder zur Datenübertragung ein.

Von der Hardware werden Zugriffe über Interrupte initiiert, die die CPU veranlassen, die laufende Bearbeitung einzufrieren und die Interrupt Service Routine (ISR) des betreffenden Treibers aufzurufen. Die Ausführung der ISR erfolgt in privilegierten Kontexten bzw. höheren Prioritäten, um eine zeitnahe Bearbeitung zu erlauben. Aktive ISRs können nicht unterbrochen werden, sodass es für einen stabilen Betrieb des Computers essenziell ist, die ISR so kurz wie möglich zu gestalten. Daher werden die Abläufe innerhalb dieser Routinen auf die zeitkritischen Kontroll- und Datenübertragungsfunktionen beschränkt, die für die korrekte Arbeitsweise der jeweiligen Hardware erforderlich sind. Die anschließende Weiterverarbeitung der Daten oder die Benachrichtigung der Anwendungsschicht erfolgt über nachgelagerte zurückgestellte Aufrufe (*Deferred Procedure Calls* (DPC), *bottom halves*), die in niedriger priorisierten Kontexten laufen und unterbrechbar sind.

Jede Unterbrechung der CPU durch einen Interrupt erfordert einen Kontextwechsel zwischen User- und Kernel-Modus. Vor dem Aufruf der ISR muss der vollständige Zustand des vor dem Interrupt aktiven Kontext (CPU-Status, MMU-Zustand, etc.) gesichert und nach dem Rücksprung rekonstruiert werden. Dieses ist mit einem nicht vernachlässigbaren Aufwand verbunden, sodass beim Entwurf von Hardware und deren Anbindung an einem Rechner die Interruptfrequenz so abzuwägen ist, dass einerseits die Reaktionszeit auf ein externes Ereignis gering ist, andererseits die Effizienz des Systems nicht durch zu häufige Kontextwechsel beeinträchtigt wird. Die maximal tolerierbare Interruptfrequenz ist sehr stark abhängig vom verwendeten System, typischerweise werden die ISR von Gerätetreibern mit einer Frequenz von etwa 100 Hz aufgerufen. Üblicherweise ist der als Zeitbasis dienende Timer Interrupt im Bereich von 1000 Hz der häufigste Interrupt in einem System und bildet die obere Grenze für den reibungslosen Betrieb des Rechners. Höhere kontinuierliche Raten machen zudem wenig Sinn, da sie dann innerhalb der kurzen Periode nicht zeitig akkurat genug abgearbeitet werden können.

Für die Latenzanalyse gehen daraus zwei relevante Aspekte hervor: erstens können kontinuierliche Datenströme nicht mit beliebiger zeitlicher Granularität zwischen Peripherie und CPU ausgetauscht werden, sondern müssen hardwareseitig gesammelt und in größeren Blöcken ausgetauscht werden. Es ergibt sich eine Pufferlatenz gemäß Formel (5.1). Zweitens ist zu beachten, dass der asynchrone und unterschiedlich priorisierte Zugriff zwischen Anwendungsschicht und Kernel einerseits und zwischen Kernel und Hardware andererseits eine Latenzquelle darstellen kann, da Daten für die Dauer der Synchronisierung im Kernel verweilen.

5.2.1.2 Verzögerungen durch Scheduling

Die zweite für NMP relevante Komponente innerhalb des Kernels ist der Scheduler, der den Ablauf quasigleichzeitiger Prozesse und Threads kontrolliert und deren Zugriff auf gemeinsam verwendete Ressourcen steuert. Latenzen ergeben sich für unsere Anforderung dabei im Scheduler bei der Zuteilung der CPU-Zeit an die auszuführenden Prozesse.

Diese Zuteilung erfolgt vereinfacht nach einem Zeitschlitzverfahren unter Berücksichtigung unterschiedlicher Prozessprioritäten. Der Scheduler teilt die CPU-Zeit in gleich große Zeitabschnitte, die als *Quantum* bezeichnet wird. Aus den Listen der auf Ausführung wartenden Prozesse wird ein Prozess anhand von Fairness- und Prioritätsabwägungen ausgewählt und für die Dauer des nächsten Quantums zur Ausführung gebracht. Für den Kontextwechsel ist eine Unterbrechung der Prozessausführung und ein Wechsel vom User- in den Kernel-Space nötig, das mit ähnlichen Aufwand für die CPU verbunden ist wie die Ausführung einer ISR zur Bearbeitung eines Interrupts. Bei der Festlegung der Länge des verwendeten Quantums muss auch für den Scheduler ein Kompromiss zwischen Effizienz und Reaktionsfähigkeit eingegangen werden. Wählt man diese zu kurz, steigt das Verhältnis von CPU-Zyklen für das Scheduling gegenüber denen für die Bearbeitung der Prozesse – die Effizienz fällt. Sind die Zeitschlitze zu breit, wird das System träge, da ein zurückgestellter Prozess zu lange auf die Zuteilung der CPU warten muss.

Bei heute gebräuchlichen Desktop Betriebssystemen liegt dieser Wert beispielsweise bei 10 ms unter WindowsXP, unter Linux werden unterschiedliche Werte unterstützt, geläufig sind Werte von 10 ms, 4 ms und 1 ms. Diese Betrachtung ist an dieser Stelle stark zusammengefasst wiedergegeben, genauere Untersuchung werden beispielsweise unter [28] durchgeführt.

Für den Betrieb von NMP ist dieser Zusammenhang für den Entwurf des Systems relevant, da der Ablauf von durch Zeitgeber gesteuerten Prozessen wesentlich von der Wahl des Quantums abhängt. Über Zeitgeber gesteuerte synchrone Prozesse können nur mit der Granularität des Quantums zur Ausführung gebracht werden, wodurch bspw. unter Windows eine mittlere Ungenauigkeit von 5 ms zu berücksichtigen ist.

5.2.2 Latenzen in Hardware und Peripherie

Die Latenzen in Systemhardware und Peripherie setzen sich zusammen aus der deterministischen Übertragungszeit der Daten und der statistischen Wartezeit beim gemeinsamen Zugriff auf ein Betriebsmittel.

Bei der hardwareseitigen Ressourcenteilung entstehen Totzeiten, die nicht vom Benutzer beeinflussbar sind. Gemeinsam genutzte Ressourcen wie Datenbusse integrieren den so genannten *Arbiter*, der die Zugriffe mehrerer Teilnehmer auf dem Bus in Hardware regelt. Mehr oder weniger ausgefeilte Mechanismen sorgen für eine gewisse Fairness bei der Verteilung, jeder Teilnehmer bekommt einen exklusiven und zeitlich befristeten Zugriff auf das Medium. Die Umsetzung dieses Arbiters ist abhängig vom Bus und der Anzahl gleichzeitiger Teilnehmer. So reicht beim Speicherbus eine Signalleitung, die den Zugriff auf den Hauptspeicher entweder der CPU oder der DMA (Direkter Speicherzugriff der Peripherie) zubilligt, während beim USB unterschiedlichste Geräte unterstützt und verschieden priorisierte Datenübertragungsklassen berücksichtigt werden. Gemeinsam ist allen Arbitern durch die Realisierung in Hardware eine geringe Wartezeit zwischen Anforderung und Zuteilung, die sich in Größenordnungen von Mikrosekunden bewegt.

Nach der Zuteilung der Ressource erfolgt der Datentransfer mit einer Übertragungsverzögerung, die sich anhand der nominellen Datentransferrate des verwendeten Busses nach unten abschätzen lässt. In Tabelle 5.1 sind Abschätzungen für die Übertragung von einem 1 KB-Datenblock über verschiedene Busse aufgeführt. So kann der Transfer von einem Kilobyte Daten über den PCI-2.0 Bus zwischen 8 μs im Burst Modus und 46 μs bei der Übertragung der Daten als einzelne Worte dauern. Solch vernachlässigbar kleine Werte sind über USB2 nur im schnellsten Modus zu erreichen, im Full- und Low-Speed Modus erhöhen sich die Übertragungsverzögerungen auf 667 bzw. 5340 μs. Bei der Abschätzung der Werte für die ATAPI-Schnittstelle ist weniger die nominelle Übertragungskapazität von 133 MBps der ATA-7 Spezifikation heranzuziehen, als die Leistungsangaben der verwendeten Festplatte. Geht man von einem Modell mit einer Dauertransferrate von 50 MBps und einer mittleren Zugriffszeit von 9 ms aus, ergibt sich eine Wartezeit von bis zu 9020 μs für den wahlfreien Zugriff auf einen Block. Ein Zugriff auf die Festplatte während der zeitkritischen Audiodatenverarbeitung birgt daher hohes Störungspotenzial und gefährdet den Betrieb von NMP.

Tabelle 5.1: Verzögerungen bei der Übertragung von 1 KB über verschiedene Busse

Bus		minimale Übertragungslatenz [μs]
PCI 2.0	Burst	7.8
	Einzeltransfer	46 (6 Clock-Cycles/word)
USB 2.0	Low-Speed (1,5 Mbps)	5340
	Full-Speed (12 Mbps)	667
	High-Speed (480 Mbps)	16.7
ATAPI	Burst	20
	Einzeltransfer	9020 (seek 9 ms)

Ergänzend zu diesen vorhersagbaren Minimalwerten sind die statistischen Zeiten bis zur Zuteilung der erforderlichen Ressourcen zu berücksichtigen. Eine genaue Abschätzung ist aufgrund der dynamischen und stark von der verwendeten Hardware abhängigen Arbitrierung nicht möglich. Als Direktive für einen störungsfreien Betrieb von NMP ist eine minimale Belastung aller verwendeten Ressourcen anzustreben.

Weil für den Benutzer praktisch keine direkten Kontrollmöglichkeiten auf die Operationen der Hardware bestehen, beschränken wir uns bei der Betrachtung dieses Aspektes auf die hier offenbarten Erkenntnisse:

Auswahl geeigneter Hardware

Die Hardware muss für NMP so gewählt werden, dass die Audiodatenströme kontinuierlich unterbrechungs- und störungsfrei bearbeitet werden können. Insbesondere ist dabei darauf zu ach-

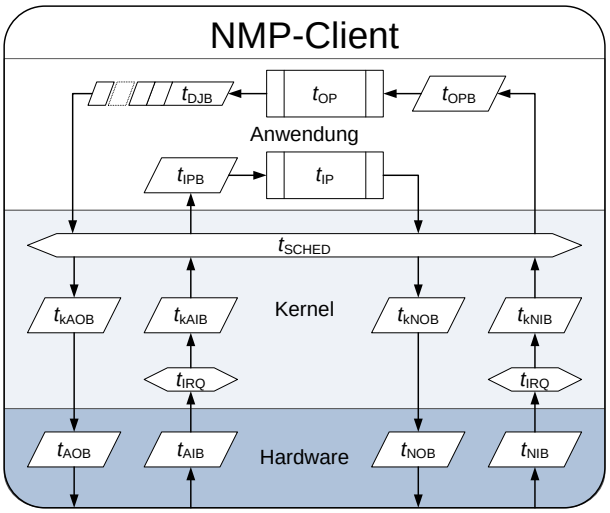

Abbildung 5.2: Daten- und Latenzpfad im NMP-Client

ten, dass keine Busse zu verwenden sind, die bereits aufgrund niedriger Datenraten zu signifikanten Übertragungsverzögerungen (wie bei USB) führen würden.

Geringe Auslastung der Ressourcen

Mit steigender Auslastung vorhandener Ressourcen steigt die Wahrscheinlichkeit, dass im Verarbeitungspfad von NMP auf die entsprechende Zuteilung gewartet werden muss. Die einzig effektive Möglichkeit zur Reduzierung des Risikos ist die Minimierung der Auslastung aller Ressourcen.

Vermeidung von Zugriffen auf langsame Hardware

Zeitintensive Zugriffe auf Hardware sind zu vermeiden, um Totzeiten auszuschließen. Das gilt insbesondere für Zugriffe, die in großen Blöcken abgearbeitet werden, wie bspw. Zugriffe auf Festplatten.

5.3 Verzögerungen in den Endsystemen

Die beiden Endsysteme in Form des NMP-Clients und des Servers unterscheiden sich grundlegend in ihrer Funktionsweise. Während der Client die Schnittstelle zum Benutzer über das Audiointerface bereitstellt, fungiert der Server als zentraler Audio-Mischer und -Synchronisierer im Netz. Die Daten- und Latenzpfade sind daher grundverschieden und werden in der folgenden Untersuchung separat betrachtet.

5.3.1 Client

Die grundlegende Funktionalität des Clients lässt sich auf die Umsetzung von zwei Datenpfaden herunterbrechen. Auf dem Produktionspfad wird das am Mikrofon aufgezeichnete und in der Audiohardware digitalisierte Audiosignal über die Netzschnittstelle zum Server versendet, auf dem entgegengesetzten Verbrauchspfad werden die vom Server zurückgesendeten Pakete an der Netzschnittstelle abgegriffen und an die Audiohardware zur Ausgabe weitergeleitet. In Abbildung 5.2 sind diese Abläufe schematisch dargestellt. Die Pfade durchlaufen in beiden Richtungen jeweils die Schichten der Hardware, des Betriebssystems (Kernel) und der Anwenderschicht.

Betrachten wir den Produktionspfad nun etwas genauer. Die Audiohardware zeichnet das Mikrofonsignal auf und digitalisiert es mit vorgegebenen zeitlichen und räumlichen Abtastraten. Diese zeit- und ortsdiskreten Werte (Samples genannt) werden nicht unmittelbar dem Betriebssystem übergeben, sondern in

Hardwarepuffern gesammelt und das Betriebssystem über einen Interrupt erst benachrichtigt, wenn ein Block vorgegebener Länge vorhanden ist. Audioparameter und Blockgröße sind innerhalb einer Sitzung konstant, sodass die Soundkarte in äquidistanten zeitlichen Abständen Datenblöcke produziert und konsumiert.

Der Gerätetreiber der Soundkarte überträgt die eingehenden Audiodaten innerhalb der Interrupt-Service-Routine (ISR) von der Hardware in Kernel-interne Puffer und schreibt meist gleichzeitig die für die Wiedergabe bestimmten Daten in den Ausgabepuffer der Audiohardware. Im Kernel werden diese Puffer in Warteschlangen verwaltet und können mit den in der Anwendungsschicht auf die Audioschnittstelle zugreifenden Prozessen ausgetauscht werden.

Der Datenpfad zwischen Netzhardware und Anwendung unterscheidet sich wesentlich von dem zwischen Soundkarte und Anwendung. Zum einen werden die Netzpakete nicht nur zwischen Hardware und Anwendung ausgetauscht, sondern im Kernel gemäß der protokollspezifischen Anforderungen verarbeitet und um Protokoll-Header erweitert. Zum anderen ist der Datentransport über die Netzschnittstelle nicht deterministisch. Ob oder wann ein zum Versenden bereitgestelltes Paket tatsächlich übertragen wird, hängt von vielen Faktoren ab. Im Idealfall ist das Übertragungsmedium immer frei und die Daten können unmittelbar mit hoher Geschwindigkeit versendet werden. In der Realität treten stochastische Verzögerungen auf, die wir als Teil des Netzpfades behandeln und in die Netzlatenz einfließen lassen, die wir im Abschnitt 5.4 genauer untersuchen. In den Endsystemen berücksichtigen wir als Verzögerung beim Zugriff auf die Netzschnittstelle lediglich die Zeiten für die Bearbeitung der Interrupte und des Schedulings.

Der Datenaustausch zwischen Kernel- und Userspace erfolgt durch zeit- oder signalgetriggerte Aufrufe von auf den Daten wartenden Routinen. Die entsprechende Benachrichtigung der Prozesse bzw. die Zuteilung gemeinsam genutzter Ressourcen erfolgt über den Scheduler, der, wie einleitend beschrieben, System abhängig ist und variierende Ausführungszeiten hat. Dieser Scheduling-Jitter wird durch die Verwendung zusätzlicher Warteschlangen im Kernel kompensiert, um zu gewährleisten, dass auch mit der langsamen Schnittstelle zur Anwendungsschicht die zeitkritische Datenein- und -Ausgabe kontinuierlich und verlustlos erfolgen kann.

Auf der Anwendungsebene findet schließlich die Bearbeitung der Audiodaten statt, die beliebig komplex und zeitaufwendig ausfallen kann. Im einfachsten Fall werden die Datenblöcke nur zwischen Audio- und Netzschnittstelle ausgetauscht. Tatsächlich sind in einem NMP-Client zusätzlich mindestens Funktionen zur Datenreduktion durch eine Encodierung der Audiodaten und die Erkennung und Verdeckung von Paketverlusten vorzusehen. Diese Datenverarbeitung verursacht zusätzliche Latenzen durch die CPU-Zeit und gegebenenfalls weiterer Pufferung.

Die grobe Abschätzung der im Client zu erwartenden Verweildauer der Audiodaten erfolgt nun anhand der in Abbildung 5.2 dargestellt Daten- und Latenzpfade. Wir gehen zunächst davon aus, dass jede Schnittstelle zur Hardware in beide Richtungen unabhängig von je einem Thread bearbeitet wird, der für jeden Zugriff auf die Hardware vom Betriebssystem benachrichtigt wird.

Für die Pufferung der am Analog-Digital-Umsetzer (ADU) digitalisierten Audiodaten in der Hardware wird die Zeit t_{AIB} gemäß Formel (5.1) benötigt. Die Soundkarte signalisiert dem Betriebssystem über einen Interrupt die Bereitstellung eines Datenpuffers, der innerhalb der ISR in den Hauptspeicher kopiert werden muss. Vom Auslösen des Interrupt-Signals am Prozessor bis zur Abarbeitung und zum Rücksprung aus der ISR vergeht eine Zeit t_{IRQ}. Im Kernel werden die eingelesenen Daten in eine Warteschlange eingereiht und stehen dem wartenden Thread nach einer Benachrichtigung zur Verfügung. Die Verweildauer der Daten im Kernel t_{kAIB} ist abhängig von der Zeit t_{SCHED} für das Scheduling. Falls der Datenaustausch zwischen Kernel- und Userspace in anderen Blockgrößen erfolgt als zwischen Hardware und Betriebssystem, gilt die in Formel (5.2) aufgeführte Berechnung der Verzögerung durch Pufferung.

In der Applikationsebene fassen wir die Latenzen für die Datenverarbeitung und für zusätzliche Puffe-

rung zusammen. Dabei wird die CPU-Zeit aller durchlaufenen Funktionsblöcke zu t_{cOP} aufsummiert, während in t_{cOPB} der in der gesamten Verarbeitungskette größte Datenblock zu einer auch hierbei nach Formel (5.2) zu berechnenden Pufferlatenz führt. Die Bearbeitung größerer Datenblöcke kann an dieser Stelle für die Umsetzung bestimmter Audiocodecs oder Fehlerkorrekturverfahren nötig werden.

Nach der erfolgten Bearbeitung der Audiodaten paketisiert die Anwendung diese und sendet sie an die Netzschnittstelle. Auf dieser Etappe kopiert das Betriebssystem das Datenpaket zunächst in den Kernel und erweitert es um die Protokoll-Header. Wie eingangs bemerkt, betrachten wir die Verweilzeiten der Daten in der Netzhardware und -Treiber als Teil der Netzlatenz, die wir in Abschnitt 5.4 genauer untersuchen.

Die Rückrichtung der vom Server empfangenen Netzpakete verläuft in umgekehrter Reihenfolge bis hin zur Übergabe der rekonstruierten Audioblöcke zur Ausgabe an die Audiohardware. Während jedoch in Senderichtung von der Soundkarte eingelesene Daten unmittelbar zum Server übermittelt werden können, ist in Empfangsrichtung wegen der variierenden Übertragungsverzögerung im Netz ein De-Jitter Puffer in Form eines Abspielpuffers vorzusehen, um eine kontinuierliche und fehlerfreie Audioausgabe zu gewährleisten. Die Länge dieses Puffers wird entsprechend der Netzcharakteristik und den Benutzervorgaben gewählt und führt auf der Applikationsebene zu einer Verzögerung von t_{cDJB}.

Damit ergibt sich für die Verweildauer der Daten im NMP-Client t_{CP} ein Wert von

$$\begin{aligned} t_{\mathrm{CP}} \quad = \quad & t_{\mathrm{AIB}} + t_{\mathrm{AOB}} \\ & + 2 \cdot t_{\mathrm{IRQ}} + 4 \cdot t_{\mathrm{SCHED}} + t_{\mathrm{kAOB}} + t_{\mathrm{kAIB}} \\ & + t_{\mathrm{cIPB}} + t_{\mathrm{cIP}} + t_{\mathrm{cOPB}} + t_{\mathrm{cOP}} + t_{\mathrm{cDJB}} \end{aligned} \tag{5.3}$$

Für die Abschätzung der unteren Schranke wird eine so detaillierte Betrachtung nicht benötigt, da nur die unvermeidbaren Verzögerungen im Idealfall relevant sind. In Abschnitt 5.3.4 wird auf diese Formel bei der Bestimmung der Verzögerung in den Endsystemen zurückgegriffen. Zunächst schauen wir jedoch etwas genauer auf das andere Endsystem.

5.3.2 Server

Der NMP-Server ist in seiner Grundfunktion für das zeitsynchrone Mischen der Audiodaten aller Teilnehmer einer Sitzung zuständig. Sein Aufbau und der Datenpfad ist schematisch in Abbildung 5.3 dargestellt. Der augenfälligste Unterschied zum Client ist das Fehlen der Schnittstelle zur Audiohardware. Die Daten durchlaufen so weniger Stationen, dadurch sinkt die Anzahl möglicher Quellen für Verzögerungen. Neben dieser prinzipbedingten Abweichung sind identische und vergleichbare Abschnitte deutlich sichtbar, die Unterschiede in der Anwendungsschicht sind marginal. Die signifikanteste Abweichung ist, dass der Server als zentraler Knoten die Daten aller teilnehmenden Clients bearbeiten muss, die Anzahl n_C der Teilnehmer geht also multiplikativ in die Verarbeitungszeit ein.

Während der Client die vom Server empfangenen Audiopakete nach einer definierten Verweildauer im De-Jitter Puffer an die Audioschnittstelle verzögert zur Ausgabe übergibt, verwaltet der Server für jeden Teilnehmer einer Sitzung einen vergleichbaren De-Jitter Puffer. Das zeitsynchron gemischte Audiopaket wird hierbei direkt nach dem Mischen verarbeitet und unmittelbar an alle Teilnehmer zurückgesendet. Die Wartezeit der Daten in diesen Puffern hängt wie beim Client von den Netzeigenschaften und von Benutzereinstellungen ab.

Für die im Server zu erwartende Latenz t_{SP} gilt somit

$$t_{\mathrm{SP}} \quad = n_C \cdot \left(t_{\mathrm{IRQ}} + 2 \cdot t_{\mathrm{SCHED}} + t_{\mathrm{sIPB}} + t_{\mathrm{sIP}} + t_{\mathrm{sOPB}} + t_{\mathrm{sOP}} \right) + t_{\mathrm{sDJB}} \tag{5.4}$$

Auch diese Abschätzung ist allgemeingültig, die bei der folgenden Abschätzung der unteren Schranke idealisiert und vereinfacht wird.

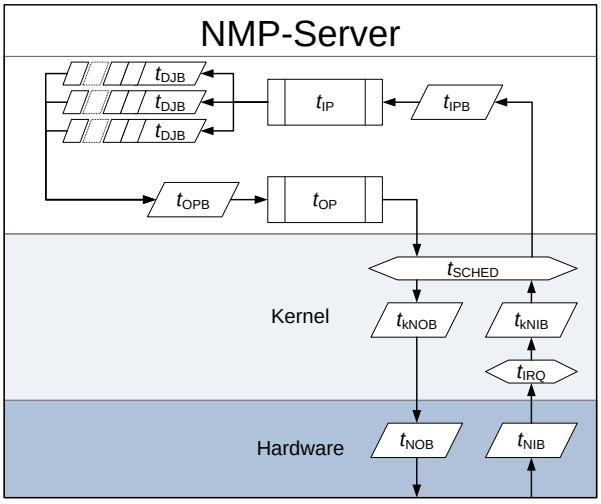

Abbildung 5.3: Daten- und Latenzpfad im NMP-Server

5.3.3 Bestimmung der Puffergranularität und der Pufferlatenz

Wir sehen, dass ein beträchtlicher Anteil der zu erwartenden Latenz durch das Puffern von Daten entsteht. Gemäß Formel (5.2) können die Pufferlatenzen dadurch minimiert werden, dass in der Anwendung durchgängig alle Puffer als Vielfache einer einheitlichen atomaren Dateneinheit verwendet wird. Geringst mögliche Latenz wird dann erreicht, wenn alle Funktionseinheiten mit gleich großen Blöcken operieren und dadurch im Pipeliningverfahren ohne zusätzliche Verzögerung gearbeitet werden kann.

In einer Audioanwendung legt die Audiohardware die Puffergranularität durch den blockbasierten Datenaustausch mit dem Betriebssystem fest. Die Audiodaten werden dabei in Abhängigkeit von der Gewählten Abtastrate D_r zunächst auf der Hardware in Abschnitte zu je B_l Samples paketisiert und anschließend als Block dem Rechner übergeben. Um die resultierende Paketisierungslatenz zu minimieren, ergeben sich laut Formel 5.1 zwei Forderungen: die Audiohardware muss einerseits den Datenaustausch mit der CPU in möglichst kleinen Quantitäten unterstützen, andererseits muss eine möglichst hohe Abtastrate verwendet werden, um die Puffer schnell zu füllen. Für beide Forderungen bestehen technische und architektonische Einschränkungen, die bei der Umsetzung zu berücksichtigen sind.

Betrachten wir zunächst einmal die Grenzen bei der Wahl der Abtastrate. Für die Aufzeichnung digitaler Audiodaten auf CDs legt der Redbook Standard (ANSI IEC-908) eine Abtastrate von 44.1 kHz fest, was bereits ausreichend ist, um alle vom menschlichen Gehör wahrnehmbaren Frequenzen verfälschungsfrei zu rekonstruieren. Im professionellen Bereich wird eher mit der aus der Verwendung beim DAT (digitales Tonband) historisch begründeten Abtastrate von 48 kHz gearbeitet. Diese Abtastfrequenz wird typischerweise von den meisten Audiokarten als Obergrenze unterstützt, sodass wir in NMP 48 kHz als Standard Abtastfrequenz verwenden.

Bei der Bestimmung der minimalen Blockgröße für den Datenaustausch zwischen Audiokarte und CPU ist dagegen kein einheitlicher Wert vorgegeben. Dieser ist einerseits abhängig von der verwendeten Hardware und trägt andererseits dem blockbasierten Zugriffsschema des Betriebssystems auf die Audiodaten Rechnung.

Technische Schranken nach unten kann es bereits durch die verwendeten Umsetzer geben, die unter Umständen selbst nur eine Datenübergabe in vorgegebenen Quantitäten erlauben. So ist der Zugriff auf die digitalisierten Daten eines Analog-Digital-Umsetzers (ADU) nicht immer sampleweise möglich, genauso benötigen bestimmte Typen von Digital-Analog-Umsetzern (DAU) eine Mindestanzahl von Samples, um ei-

ne interpolierende Wandlung durchzuführen. Als Richtwert seien hier die Angaben von CirrusLogic für den *Group Delay* des CS4398 von 9.4 Fs (also dem 9.4 Fachen der Sampledauer) bzw. von 12 Fs für den CS5381 genannt, die bei einer Samplerate von 48 kHz zu einer Verzögerung von 0.2 bzw. 0.25 μs führen.

Unabhängig von dieser Einschränkung werden darüber hinaus deutlich größere Puffer auf der Audiohardware verwendet, um den Erfordernissen beim Datenaustausch mit dem Betriebssystem zu genügen. Gemäß obiger Betrachtung sinnvoller Interruptfrequenzen für Gerätetreiber herrscht bei den Herstellern Konsens darüber, dass heutige Rechnersysteme nicht mehr als etwa 500 Interrupte pro Sekunde sicher verarbeiten können. Dementsprechend wird die Länge der auf der Hardware zu verwendenden Puffergrößen dimensioniert. In Tabelle 5.2 sind exemplarisch die minimalen Blockgrößen einiger Audiokarten und die resultierenden Paketisierungslatenzen bei einer Abtastrate von 48 kHz aufgeführt.

Tabelle 5.2: Minimale Blockgröße bei 48 kHz Abtastrate

Hardware	Blockgröße [Samples]	Latenz [ms]
AC97 onboard	128	2.66
Sound Blaster Audigy2	192	4.00
Echoaudio INDIGO io	96	2.00
EMU-1616m	128	2.66
iMac onboard	128	2.66

Die aufgeführten Werte stellen die untere Schranke dar, die nur unter optimalen Bedingungen kontinuierliches und fehlerfreies Streaming ermöglicht. Die tatsächlich verwendeten Puffergrößen werden von Treiber, Betriebssystem und Anwendungssoftware angepasst. Unsere Untersuchung verfügbarer Audiohardware hat gezeigt, dass mit wenigen Ausnahmen eine minimale Puffergröße von 128 Samples bei einer Abtastfrequenz von 48 kHz unterstützt wird, und wir in diesem Fall von einer atomaren Blockverzögerung von t_φ =2.66 ms rechnen können.

In Abhängigkeit von der Komplexität und Auslastung des verwendeten Systems und damit der Trägheit bei der Bearbeitung von Hardwareinterrupts kann es erforderlich sein, die Puffergröße zu erhöhen oder mehrere Audioblöcke zunächst im Kernel zu sammeln, bevor sie zwischen Hardware und Anwendung ausgetauscht werden.

Eine auf minimale Latenz hin optimierte Audiokarte folgt einem bewährten Prinzip, das von den meisten Herstellern professioneller Produkte (wie zum Beispiel Steinbergs ASIO [89]) verwendet wird. Es beruht auf der Verwendung von Pufferhälften vorgegebener Größe mit der Pufferverzögerung von jeweils t_φ für Ein- und Ausgaberichtung. Auf die jeweils aktiven Front-Puffer greifen dabei der DAU und der ADU der Audiokarte zu, während die CPU die digitalisierten Werte aus dem Back-Puffer liest und die auszugebenden Samples dorthin schreibt. Der minimale zeitliche Abstand zwischen der Ein- und Ausgabe eines Samples zwischen Kernel und Audiohardware beträgt daher zwei Pufferverzögerungen, also $2 \cdot t_\varphi$.

Der Datenaustausch zwischen der Hardware und dem Kernel erfolgt synchron in periodischen Abständen von t_φ. Bei einem vorgegebenen Füllstand der Puffer löst die Karte einen Interrupt aus und verursacht eine zeitnahe Abarbeitung der entsprechenden ISR. Um einen Über- bzw. Unterlauf der Puffer zu verhindern, werden in der ISR lediglich die Back-Puffer auf der Audiokarte mit den Kernel-Puffern aktualisiert. Die Weitergabe der Daten an die Prozesse in der Anwendungsschicht erfolgt anschließend in nachgelagerten DPCs. Dieses Schema bedingt, dass die Eingangsdaten von der Audiokarte unmittelbar in der DPC ausgelesen werden können, während in Ausgaberichtung bereits der nächste auszugebende Audioblock in den Kernel zu schreiben ist, um beim nächsten Interrupt unmittelbar in der ISR an die Hardware geschrieben werden zu können. Daher sind die am Mikrofon digitalisierten Samples beim Eintreffen in die Anwendungsschicht

mindestens t_φ alt, während das am Lautsprecher abgespielte Signal von der Anwendung mindestens $2 \cdot t_\varphi$ vorher von der Anwendung ausgegeben wurde.

Für die analytisch bestimmten Größen gilt damit:

- $D_r = 48\,\frac{Samples}{s}, B_l = 128\,Samples \quad \Rightarrow t_\varphi = 2.66\,ms$
- $t_{\mathrm{AIB}} = t_{\mathrm{AOB}} = t_{\mathrm{kAOB}} = t_\varphi$
- $t_{\mathrm{kAIB}} = 0$

5.3.4 Abschätzung der minimalen Latenz in den Endsystemen

Die Bestimmung einer unteren Schranke für die in den Endsystemen zu erwartende Verzögerung ist für die Abschätzung des potenziellen Einsatzradius unseres NMP-Systems essenziell. Basierend auf unserer Analyse und den durch technische Aspekte ermittelten Vorgaben hinsichtlich Puffergranularität und Blockverzögerung ermitteln wir nun die unvermeidbar in den Endsystemen auftretenden Latenzen.

Folgende Idealisierungen nehmen wir für die Abschätzung an:

Ideales Netz

Das Netz wird idealisiert durch eine direkte Verbindung zwischen Client und Server derart angenommen, dass wir eine unbegrenzte Übertragungskapazität annehmen und von einer verzögerungsfreien Übertragung ausgehen. Das ist legitim, da die realen Latenzen im Netz gesondert in Abschnitt 5.4 behandelt werden. Durch diese Idealisierung können wir die Verweildauer der Netzpakete im Kernel t_{kNOB} und t_{kNIB} und auf der Netzschnittstelle t_{NOB} und t_{NIB} aus der Betrachtung ausklammern, da wir davon ausgehen, dass das Medium immer frei ist und Netzpakete nicht zwischengespeichert werden müssen. Das am Server empfangene Paketmuster entspricht danach exakt dem vom Client versendeten.

Latenzoptimaler Zugriff auf Audiodaten

Wir nehmen ein auf geringe Latenz hin optimiertes Schema beim Austausch der Audiodaten zwischen Audiohardware und Betriebssystem an, wie im vorherigen Abschnitt beschrieben. Dabei wird mit einer Samplerate von 48 kHz und einer Audioblockgröße von 128 Samples gearbeitet, was zu einer Blockverzögerung von t_φ =2.66 ms führt. Nach dem beschriebenen Schema gilt weiterhin $t_{\mathrm{AIB}} = t_{\mathrm{AOB}} = t_{\mathrm{kAOB}} = t_\varphi$ und $t_{\mathrm{kAIB}} = 0$.

Keine zusätzliche Verzögerung durch Pufferung

Wir nehmen an, dass jede Funktionseinheit mit der Blockgröße von 128 Samples operiert und dadurch ein Pipelining durchgeführt wird. Diese Forderung impliziert, dass jeweils ein Audiopaket einzeln über das Netz versendet wird.

Minimale Datenverarbeitung

Auf Applikationsebene nehmen wir latenzoptimierte Abläufe an. Um dabei zusätzliche Pufferung zu vermeiden und eine Minimierung der Komplexität und damit der CPU-Zeit zu erreichen, berücksichtigen wir auf dieser Ebene nur die grundlegenden Funktionalitäten, die für die Endsysteme erforderlich sind. Beim Client reduziert sich die Verarbeitung damit auf das Kopieren der Audiopakete zwischen der Audio- und der Netzschnittstelle. Der Server muss dagegen als minimale Funktion das zeitsynchrone Mischen von Audiopaketen unterstützen. Mit dem Wegfall zusätzlicher Puffer auf der Anwendungsebene gilt $t_{\mathrm{cIPB}} = t_{\mathrm{cOPB}} = t_{\mathrm{sIPB}} = t_{\mathrm{sOPB}} = 0$. Bei minimaler Komplexität der Datenverarbeitung nehmen wir außerdem an, dass die Ausführungszeiten auf dieser Ebene beim Client und Server in Vergleich zu t_φ vernachlässigbar klein sind, formal gilt $t_{\mathrm{cIP}} + t_{\mathrm{cOP}} + t_{\mathrm{sIP}} + t_{\mathrm{sOP}} \ll t_\varphi$.

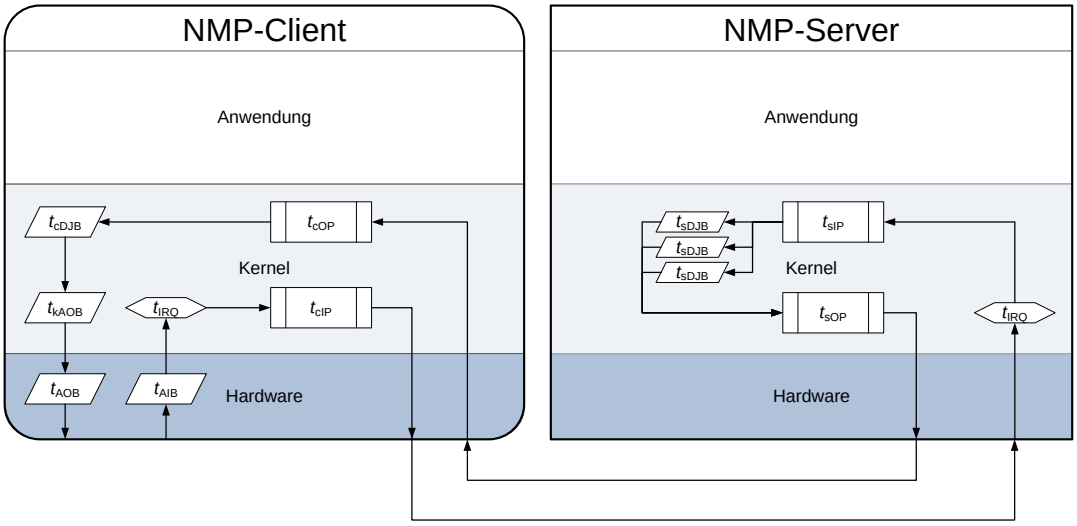

Abbildung 5.4: Idealisierter Daten- und Latenzpfad in den Endsystemen

Kein Scheduling-Overhead

Für die Minimierung der im Betriebssystem durch Prozess- und Threadscheduling entstehenden Latenzen und Jitter nehmen wir eine vollständige Realisierung von NMP-Client und Server im Kernel an. Dadurch entfallen die Zeiten für t_{SCHED} in den Endsystemen.

Ein derartiger idealisierter Aufbau von NMP ist in Abbildung 5.4 dargestellt. Ausgehend von der Abschätzung der unteren Latenzschranke in Formel (5.4) können wir für den Server mit $t_{\mathrm{sIPB}} = t_{\mathrm{sOPB}} = t_{\mathrm{SCHED}} = 0$ wie folgt vereinfachen:

$$\begin{aligned} t_{\mathrm{SP}} &= t_{\mathrm{sDJB}} + n_C \cdot \left(t_{\mathrm{IRQ}} + t_{\mathrm{sIP}} + t_{\mathrm{sOP}}\right) \\ &= t_{\mathrm{sDJB}} + t_{\mathrm{SPP}} \end{aligned}$$

Als t_{SPP} haben wir alle Verzögerungen zusammengefasst, die durch die aktive Verarbeitung im Server erfolgt. Entscheidend für die Bestimmung der minimalen Verweildauer der Daten im Server ist offenbar die erforderliche Zeit, die Audiopakete in den De-Jitter Puffer warten müssen, bis die zu einem Zeitpunkt gehörenden Pakete von allen teilnehmenden Clients eintreffen. In einem Aufbau mit den genannten Idealisierungen gilt, dass die Audiopakete jedes Clients in zeitlich konstanten Abständen eintreffen. Das bedingt auch, dass die relativen zeitlichen Abstände der Empfangszeitpunkte unterschiedlicher Clients untereinander konstant sind.

Innerhalb des Paketeintreffmusters ist ein Synchronisierungsintervall zu wählen, in dem das Eintreffen des frühesten und des spätesten zusammen gehörenden Paketes am kürzesten auseinander liegen. Im günstigsten Fall treffen die Pakete aller Clients gleichzeitig beim Server ein und können sofort gemischt und zurückgesendet werden. Andernfalls wird innerhalb des Paketeintreffmusters die größte Lücke gesucht und die Synchronisation ab dem ersten folgenden Paket gestartet. Im ungünstigsten Fall treffen die Pakete aller Clients in jeweils gleichen Abständen zueinander am Server ein. Dann wird unabhängig von der Wahl des Synchronisationsstartzeitpunktes zwischen dem ersten und dem letzten zu synchronisierenden Audiopaket $t_{\mathrm{sDJB}} = \frac{n_C - 1}{n_C} \cdot t_\varphi$ gewartet werden. Es gilt also

$$t_{\mathrm{SP}} \quad \leq \quad \frac{n_C - 1}{n_C} \cdot t_\varphi + t_{\mathrm{SPP}}$$

Für den Client gilt gilt gemäß (5.3) mit $t_{AIB} = t_{AOB} = t_{kAOB} = t_\varphi$ und $t_{kAIB} = t_{cIPB} = t_{cOPB} = t_{SCHED} = 0$ analog die Vereinfachung

$$\begin{aligned} t_{CP} &= t_{AIB} + t_{AOB} + t_{IRQ} + t_{kAOB} + t_{cIP} + t_{cOP} + t_{cDJB} \\ &= 3 \cdot t_\varphi + t_{cDJB} + \left(t_{IRQ} + t_{cIP} + t_{cOP}\right) \\ &= 3 \cdot t_\varphi + t_{cDJB} + t_{CPP} \end{aligned}$$

Auch hier sind unter t_{CPP} alle Zeiten für die Verarbeitung im Client zusammengefasst. Neben der Gesamtverzögerung von $3 * t_\varphi$ durch Pufferung ist auch hier die Verweildauer im De-Jitter Puffer des Clients eine wesentliche Größe. Anders als im Server, der asynchron arbeitet, wird der Client aus der ISR der Audiohardware ausgeführt und arbeitet damit synchron. Das vereinfacht die Bestimmung von t_{cDJB} dahin gehend, dass wir nur Vielfache von t_φ als Wertemenge annehmen können. Ein De-Jitter Puffer am Client wäre nur dann nicht erforderlich, wenn das vom Server gesendete Audiopaket zeitgleich mit dem Auslösen der Audio-ISR am Client eintrifft und sofort für die Ausgabe vorbereitet werden kann. In der Praxis wird ein solcher Zustand wegen der Asynchronität der Endsysteme untereinander nicht erreicht, daher muss mindestens ein Audiopaket im De-Jitter Puffer im Client zur Synchronisation vorgehalten werden, es gilt $t_{cDJB} = t_\varphi$.

Damit können wir die auf den gesamten Latenzpfad zwischen NMP-Client und Server, wie er in Abbildung 5.4 idealisiert dargestellt ist, entstehende Verzögerung abschätzen als

$$\begin{aligned} t_{SP} + t_{CP} &= (t_{sDJB} + t_{SPP}) + (3 \cdot t_\varphi + t_{cDJB} + t_{CPP}) \\ &= (\frac{(n_C - 1)\, t_\varphi}{n_C} + t_{SPP}) + (4 \cdot t_\varphi + t_{CPP}) \\ &= (4 + \frac{(n_C - 1)}{n_C})\, t_\varphi + t_{SPP} + t_{CPP} \\ &= 5\, t_\varphi + (t_{SPP} + t_{CPP} - \frac{t_\varphi}{n_C}) \end{aligned}$$

Gehen wir weiter davon aus, dass t_{IRQ} sehr klein ist und die Bearbeitungszeiten t_{cIP}, t_{cOP}, t_{sIP} und t_{sOP} minimiert werden, wird auch die Summe der Verarbeitungszeiten in den Endsystemen in Relation zu t_φ vernachlässigbar klein. Unter der Annahme, dass $(t_{SPP} + t_{CPP}) \leq \frac{t_\varphi}{n_C}$ gilt, können wir abschließend die untere Schranke für die Verzögerung in den Endsystemen abschätzen als:

$$\begin{aligned} t_{SP} + t_{CP} &= 5\, t_\varphi + (t_{SPP} + t_{CPP} - \frac{t_\varphi}{n_C}) \\ &\leq \underline{\underline{5\, t_\varphi}} \end{aligned} \tag{5.5}$$

Mit einen Wert von t_φ =2.66 ms liegt die untere Schranke der in den Endsystemen auftretenden Verzögerungen bei 13.3 ms. Von der in einem NMP-System tolerierbaren Gesamtlatenz von 30 ms geht somit nahezu die Hälfte innerhalb der Endsysteme verloren, falls diese latenzoptimal umgesetzt werden können. Für die Verzögerung im Netz steht im Idealfall ein Budget von maximal 16.6 ms zur Verfügung.

Wo Latenzen im Netz entstehen, wird im kommenden Abschnitt betrachtet, bevor abschließend eine Vorhersage über den potentiellen Einsatzradius von NMP gemacht werden kann.

5.4 Verzögerungen im Netz

Die Übertragung der Daten über das IP-Netz vom Absender zum Empfänger erfolgt über eine variable Anzahl von Zwischensystemen (Routern), wie in Abbildung 5.5 veranschaulicht. Ein Datenpaket wird von einem

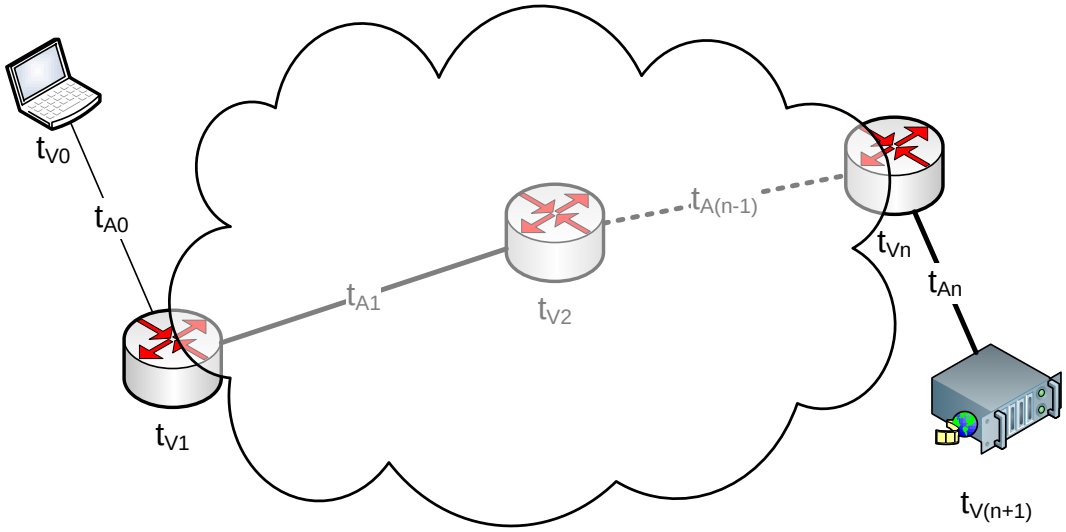

Abbildung 5.5: Übertragungsverzögerung im Netz

Router empfangen, eventuell bearbeitet und anschließend solange zwischengespeichert, bis das Übertragungsmedium zum nächsten Knoten frei ist. Die Übertragung über das Medium erfolgt mit einer bestimmten Datenrate, die physikalische Länge des Mediums bestimmt die Ausbreitungsverzögerung.

Auf dem Übertragungspfad sind demnach folgende Arten der Verzögerung zu unterscheiden:

Paketisierung

Verzögerungen entstehen beim Paketisieren dadurch, dass Daten zunächst gesammelt werden müssen, bevor sie zu einem Datenpaket zusammengefasst werden können. Diese Latenz spielt in unserem Fall keine Rolle, da sie bereits beim Zugriff auf die Audioschnittstelle berücksichtigt wurde und jedes Audiopaket einzeln als Datenpaket versendet wird. Erfordern Komponenten oder Protokolle bei der Übertragung eine Fragmentierung oder eine Zusammenfassung von Paketen, gilt für die Berechnung der Pufferverzögerung Formel (5.2).

Übertragungsverzögerung t_{Rt}

Der Zugriff auf das Medium erfolgt mit einer definierten Datenrate r, die in Bits pro Sekunde angegeben wird. Ein Paket der Länge s erfährt beim Versenden eine Übertragungsverzögerung von $t_{\mathrm{Rt}} = \frac{s}{r}$. So benötigt das Versenden eines Netzpaketes der Länge 500 Bytes über eine Fast-Ethernet Verbindung mindestens $\frac{500 \cdot 8bit}{100 \cdot 10^6 bit/s} = 40\mu s$. Jeder Knoten auf dem Netzpfad kann das ankommende Paket erst nach vollständigem Empfang weiterleiten (Store-and-Forward), sodass eine Übertragungsverzögerung in jedem Knoten entsteht.

Ausbreitungsverzögerung t_{P}

Die Ausbreitungsgeschwindigkeit der Information im Medium ist endlich, sie wird nach oben von der Lichtgeschwindigkeit c begrenzt und ist vom verwendeten Material abhängig. Als Richtwert für die Ausbreitungsgeschwindigkeit elektromagnetischer Wellen in drahtgebundenen Netzen nehmen wir vereinfachend einen Wert von $t_{\mathrm{P}} = \frac{2}{3}c \approx 200\frac{km}{ms}$ an.

Bearbeitungszeit t_{Rr}

Sobald ein Paket am Router empfangen wurde, fällt Bearbeitungszeit für die Fehlerprüfung und das Routing an. Die Kontrolle wird benötigt, um fehlerhaft empfangene Pakete nicht unnötig weiterzuleiten. Sowohl das Routen als auch die Berechnung der Prüfsummen können bereits erfolgen, während das Paket eingelesen wird, sodass die zusätzliche Bearbeitungszeit nach dem Eintreffen des letzten

Bytes gering ist. Typische Werte für die Bearbeitungszeit liegen im Mikrosekundenbereich, auf geringe Latenz optimierte Core-Router bleiben teilweise unter einer Mikrosekunde, wodurch die Bearbeitungszeit in Relation zu der Übertragungsverzögerung vernachlässigbar klein wird.

Queuing t_{Rq}

Kann ein Paket nicht unmittelbar weiter geleitet werden, weil die ausgehende Leitung überlastet ist, muss es im Router zwischengespeichert werden. Ohne Einsatz von speziellen Scheduling-Verfahren erfolgt die Abarbeitung der eingereihten Pakete nach dem FIFO-Prinzip, die Verweildauer der Pakete ist also abhängig von der Anzahl der beim Eintreffen bereits auf den entsprechenden Link wartenden Pakete. Der Füllstand der Warteschlangen ist von außen nicht feststellbar, die Verweildauer eines Paketes in den Queues lässt sich daher nicht vorhersagen. Sie kann bestenfalls Null sein, wenn die ausgehende Leitung unmittelbar nach dem Eintreffen frei ist, andernfalls beträgt sie mindestens die Übertragungsverzögerung für alle älteren vorgehaltenen Pakete. Bei Überlast werden ankommende Pakete spätestens dann verworfen, wenn die internen Puffer vollgelaufen sind, teilweise geschieht das zur Stauvermeidung bereits vorher (*Random Early Detection*, RED [15]).

Das Queuing ist damit die wesentliche Komponente, die die Unvorhersagbarkeit der Netzverzögerung ausmacht. Alle anderen Werte wie Leitungslängen, Verarbeitungszeiten oder Datenraten zwischen den Knoten können wegen ihres statischen Charakters bestimmt werden. Das in Abhängigkeit vom Füllstatus dynamisch durchgeführte Zwischenspeichern oder Verwerfen von Paketen begründet hingegen die nicht deterministische Eigenschaft der Netzverzögerung.

Betrachten wir nun die zu erwartende Netzverzögerung in dem in Abbildung 5.5 dargestellten Netz auf dem Pfad $v_0, v_1, \ldots, v_{n+1}$. Der Sender v_0 sendet die Daten zum Empfänger v_{n+1} über n Zwischenstationen (Hops), wobei die insgesamt (n+1) Kanten a_0 bis a_n durchlaufen werden. Formal setzt sich die Netzlatenz also zusammen aus

$$t_{\mathrm{N}} = \sum_{i=0}^{n} t_{\mathrm{A_i}} + \sum_{i=0}^{n} t_{\mathrm{V_i}} = \sum_{i=0}^{n} t_{\mathrm{P_i}} + \sum_{i=0}^{n} \left(t_{\mathrm{Rt_i}} + t_{\mathrm{Rr_i}} + t_{\mathrm{Rq_i}} \right) \tag{5.6}$$

Wir nehmen idealerweise an, dass die versendeten Pakete auf ihren Weg weder fragmentiert noch zusammengefasst werden. Weiterhin gehen wir während einer NMP Session von einer konstanten Route zwischen Quelle und Ziel aus. Das ist legitim, weil ein Neuaufbau der Routen innerhalb der Laufzeit typischer NMP Sitzungen von etwa 30 Minuten unwahrscheinlich ist und im Eintrittsfall ohnehin eine Re-Initialisierung des NMP-Systems erfordert, da dadurch eine Modifikation grundlegende Betriebsparameter erfolgt.

Auf den Kanten sind die Ausbreitungsverzögerungen der einzelnen Teilabschnitte a_0 bis a_n zu berücksichtigen. Wir nehmen an, dass die Materialien sich im Bezug auf die Ausbreitungsgeschwindigkeit nicht signifikant unterscheiden und diese für alle den gleichen Wert von $v \approx 200 \frac{km}{ms}$ hat. Die Ausbreitungsverzögerung vom Quell- zum Zielrechner ist dann die Summe der Werte für die einzelnen Verbindungen

$$t_{\mathrm{A}} = \sum_{i=0}^{n} t_{Ai} = \sum_{i=0}^{n} \frac{d_i}{v} = \frac{d}{v} \tag{5.7}$$

Ist die Netzdistanz d zwischen Quelle und Ziel bekannt, kann die Ausbreitungsverzögerung unmittelbar berechnet werden, andernfalls liefert die geografische Entfernung die untere Schranke für diese Größe.

Die Übertragungsverzögerung tritt auf allen Teilstrecken der Verbindung auf, die abhängig ist von der jeweiligen Datenrate zwischen den betreffenden beiden Knoten. Üblicherweise verläuft der Pfad von einem Zugangsnetz zum anderen über Kernnetze, sodass die Datenrate nicht durchgehend konstant. Die Übertragungslatenz t_{Rt} eines Paketes der Größe s ist dann

$$t_{\mathrm{Rt}} = \sum_{i=0}^{n} t_{\mathrm{Rt_i}} = \sum_{i=0}^{n} \frac{s}{r_i} = s \cdot \sum_{i=0}^{n} \frac{1}{r_i} \tag{5.8}$$

Die Werte für die Bearbeitungszeit in den Routern variieren in Abhängigkeit von der eingesetzten Hardware und sind, ohne interne Kenntnis darüber nicht zu ermitteln.

Bei der Betrachtung der Verzögerungen durch Queuing greift man auf Modelle zurück, die anhand statistischer Annahmen über den Datenverkehr eine Voraussage über die Verweildauer der Datenpakete in den Warteschlangen ermöglichen. In einem System ist der Füllstand der Puffer abhängig von den Kapazitäten der Ein- und Ausgangsleistungen und deren Auslastung, sowie vom Art und Umfang der zu bearbeitenden Daten. Kann für einen Router analytisch oder empirisch das zu bearbeitende Paketmuster und das Datenaufkommen ermittelt werden, liefern solche Modelle eine zuverlässige statistische Abschätzung für die Verzögerung im Router.

Diese modellbasierte Untersuchung, wie sie von bspw. von Chung oder Rahmani in [21, 74] durchgeführt werden, können herangezogen werden, wenn für eine Route umfangreiche Statistiken und technische Details der Netzhardware entlang der untersuchten Route vorliegen. Sie können dann als sinnvolles Werkzeug bei der Dimensionierung der Infrastruktur herangezogen werden, die ein langfristiges Datenaufkommen bewältigen soll. Kurzfristige Aussagen, wie sie für NMP bei der Einschätzung der Netzeigenschaften für die laufende Sitzung (und damit für die nächsten etwa 30 Minuten) benötigt werden, sind damit nicht möglich.

Einen anderen Ansatz, die Verweildauer von Netzpaketen in Routern abzuschätzen, stellt das aktive Messen an Routern im regulären Betrieb dar, wie es unter anderem von Papagiannaki in [66] durchgeführt wurde. An einem Core-Router des Sprint-Backbones wurden dabei über längere Zeiträume die Verzögerungen für das Routen von Paketen ermittelt und die Wahrscheinlichkeit für die Verweildauer der Daten empirisch aufgestellt. Die für uns relevante Schlussfolgerung dieser Untersuchungen ist die, dass 99% aller Pakete im Backbone um weniger als 1 ms pro Router verzögert werden. Unter Berücksichtigung des technischen Fortschritts seit diesen in den Jahren 2001 und 2002 durchgeführten Messungen nehmen wir im Weiteren einen Wert von $t_{\mathrm{R}} = 0.5\,ms$ als Abschätzung für die mittlere Verzögerung von Netzpaketen im Router an, die sich aus Bearbeitungszeit t_{Rr} und der Zeit für das Queueing t_{Rq} zusammensetzt.

5.5 Bestimmung des theoretischen Anwendungsradius von NMP

Als Fazit der obigen Latenzanalyse stellen wir fest, dass die Verzögerungen sich sowohl in den Endsystemen als auch im Netz aus konstanten und variablen Anteilen zusammensetzen. Die konstanten Anteile geben dabei die untere Schranke in einem idealen Szenario vor, die variablen Anteile stellen die in realer Umgebung zusätzlich zu kompensierenden Verzögerungen dar.

Hier interessieren wir uns für den theoretischen maximalen Anwendungsradius von NMP und gehen somit von Idealbedingungen aus. Die minimale Verweildauer der Daten in den Endsystemen wurde mit $t_{\mathrm{ES}} = 5 \cdot t_{\varphi}$ bestimmt, für die Netzverzögerung greifen wir auf die Abschätzung zurück, wonach die Verzögerung durch einen Router unter einer halben Millisekunde liegt. Damit liegt die Netzverzögerung eines Audiopaketes für den Hin- und Rückweg vom Client zum Server über n_{R} Router

$$t_{\mathrm{N}} = 2 \cdot (t_{\mathrm{P}} + n_{\mathrm{R}} \cdot (t_{\mathrm{R}} + t_{\mathrm{Rt}})) \tag{5.9}$$

Mit Vorgriff auf spätere Ausführungen verwenden wir beim Audioformat eine zeitliche Abtastrate von 48 kHz bei 16 bpS Stereo. Mit einer konstanten Blockgröße von 128 Samples hat jedes Netzpaket 512 Bytes Nutzdaten. Zusammen mit den Protokoll-Overhead (26 Byte Ethernet, 20 IP, 8 UDP, 12 RTP) sind 578 Bytes pro Paket zu übertragen. Die Paketrate beträgt 375 Pakete pro Sekunde, gleichbedeutend mit t_{φ} =2.66 ms und einer konstanten Brutto-Datenrate von 1.734 Mbps.

Die minimale Verzögerung in den Endsystemen wird allein von t_{φ} bestimmt und beträgt mit diesem Wert t_{ES} =13.3 ms. Die Netzlatenz ist hingegen von der Anzahl der Hops, der Netzdistanz vom Client zum

Server und den Übertragungsgeschwindigkeiten zwischen den Knoten abhängig. Die obere Schranke für den Anwendungsradius wird durch die Ausbreitungsgeschwindigkeit des Signals dadurch vorgegeben, dass die Daten innerhalb der verbleibenden 16.6 ms vom Client zum Server und zurückübertragen werden müssen. Bei einer angenommenen Ausbreitungsgeschwindigkeit von 200 $\frac{km}{ms}$ darf der Client im Netz somit vom Server nicht weiter als etwa 1600 km entfernt sein.

Diesen Radius verringern Übertragungsverzögerung und Verarbeitung im Router zusätzlich, wie die Betrachtung folgender beiden Szenarien verdeutlicht.

Das erste Szenario spielt sich innerhalb des Deutschen Forschungsnetzes (DFN) ab, an dem die meisten deutschen Forschungseinrichtungen und Universitäten über das X-Win angeschlossen sind. Im X-Win nehmen wir eine Verbindung der Kernrouter untereinander über Gigabit-Ethernet mit 1000 Mbps an, im Universitätsnetz wird FastEthernet mit 100 Mbps verwendet. Die Verarbeitungszeiten in den Routern werden mit jeweils 0.5 ms pro Router abgeschätzt. Für den zu betrachtenden Aufbau ist der NMP-Server am IBR aufgestellt, ein Client ist von einer anderen Universität angebunden. Zu berücksichtigen ist, dass die Endsysteme über je zwei Hops mit einer jeweiligen Netzdistanz von 50 km mit dem X-Win verbunden sind und innerhalb des X-Win ein Router etwa alle 50 km zu durchlaufen ist.

Die Übertragungsverzögerung für ein Paket der Länge 578 Bytes bei FastEthernet beträgt $\frac{578 \cdot 8\mu s}{100} = 46.2\mu s$, bei GigabitEthernet sind es entsprechend 4.6 μs. Für die Teilstrecken zwischen den Endsystemen und dem X-Win fallen für die einfache Richtung an

$$\begin{aligned} 4 \cdot ((t_{\mathrm{R}} + t_{\mathrm{Rt}})) + t_{\mathrm{P}} &\leq 4 \cdot (46.24 + 500)\,\mu s + 100km \cdot \frac{1ms}{200km} \\ &= 2185\mu s + 500\mu s = 2685\mu s \end{aligned}$$

Für jede zusätzliche Teilstrecke im X-Win sind je Router knapp 505 μs Verarbeitungsverzögerung und 250μs für die Signalausbreitung einzuplanen. Damit lässt sich der Anwendungsradius von NMP in diesem Szenario über die Anzahl zusätzlicher Router n_R bestimmen mit

$$\begin{aligned} 16.6\text{ms} &\geq 2 \cdot \left(2685\mu s + n_R \cdot 755\mu s\right) \\ \Leftrightarrow \qquad 7 &\geq n_R \end{aligned} \tag{5.10}$$

Damit beträgt der Anwendungsradius unter Berücksichtigung der Verzögerungen in den Routern $(7 \cdot 50km + 100km) = 450km$.

Das zweite Szenario bildet einen Aufbau ab, der nicht vollständig innerhalb des DFN liegt. Darin steht der Server wie gehabt im IBR, während der Client ein privater Anschluss im Zugangsnetz eines kommerziellen Anbieters ist und über symmetrisches DSL mit je 2 Mbps für Up- und Downloadrichtung angebunden ist. Die Übertragungsverzögerungen erhöhen sich bei dieser geringen Datenrate in der *letzten Meile* auf 2312 μs. Unter der Annahme, dass sich die Übertragung im Kernnetz nicht wesentlich von der im DFN unterscheidet, verringert sich durch die erhöhte Übertragungsverzögerung beim Client der Anwendungsradius deutlich auf knapp 300 km.

5.6 Praktische Umsetzung eines latenzoptimierten NMP-Systems

Die wesentliche Anforderung bei der Umsetzung von NMP ist die Minimierung der Gesamtverzögerung.

Da wir praktisch keinen Einfluss auf die Verzögerung im Netz haben, müssen wir uns darauf beschränken, die Latenzen in den Endsystemen zu minimieren. Ziel dabei ist die Realisierung der analytisch bestimmten unteren Schranke.

In der Analyse haben wir die Quellen für Verzögerungen in den Endsystemen klassifiziert und bereits erste Anforderungen für die Einhaltung der unteren Schranke aufgestellt. Wir konnten die Abschätzung dabei

nur vornehmen, indem wir in Abschnitt 5.3 einige Idealisierung beim Systemverhalten angenommen. Wie diese praktisch umzusetzen sind, soll in diesem Abschnitt erläutert werden.

5.6.1 Audioformat und -Blockgröße

Im Exkurs über generelle Arbeitsplatzrechner und Betriebssysteme in Abschnitt 5.2 sind die technischen Vorgaben bei der Wahl des Audiodatenformates in der Hardware und der Blockgröße für den Datenaustausch zwischen Hardware und Betriebssystem deutlich geworden. Als Konsequenz haben wir für die Analyse angenommen, dass wir mit einer Abtastrate von 48 kHz arbeiten und Blöcke zu je t_φ =2.66 ms ausgetauscht werden. Dieser Konsens stellt die hinsichtlich geringer Latenz optimale Parametrisierung der Audiohardware dar, die innerhalb der untersuchten Audiokarten breite Unterstützung findet. Daneben beeinflusst die Wahl des Audiodatenformates auch andere Größen, von denen für uns das Datenaufkommen und die Audioqualität relevant sind.

Bei der Digitalisierung analoger Audiosignale bestimmt die Wahl der räumlichen und zeitlichen Abtastraten die Auflösung und den Wertebereich der digitalen Werte und damit die Abweichung zwischen dem kontinuierlichen und dem digitalisierten Signal. Bei der Parametrisierung der räumlichen Auflösung erlauben die technischen Vorgaben kaum Wahlmöglichkeiten: praktisch alle gängigen Soundkarten unterstützen acht und 16 bpS (Bits pro Sample), professionelle Karten bieten eine Auflösung von 24 oder 32 bpS. Da acht bpS nur für Sprache ausreichend sind und auf der anderen Seite eine höhere Genauigkeit als 16 bpS in einem NMP Umfeld keine Verbesserung der Audioqualität bringt und die Wahl geeigneter Hardware einschränkt, ist eine räumliche Auflösung von 16 bpS die einzig sinnvolle Wahl.

Der Wertebereich für die zeitliche Auflösung ist technisch auf Werte zwischen acht und 48 kHz bei gewöhnlicher Audiohardware beschränkt, Produkte aus dem Studiobereich unterstützen darüber hinaus auch Abtastraten von 96 und 192 kHz. In Anbetracht einer für die digitale Audio-CD gewählten Abtastrate von 44.1 kHz gilt auch hier, dass höhere Werte keinen wahrnehmbaren Qualitätsgewinn in einer NMP-Anwendung bringen. Nach unten ist die Wahl geeigneter Werte von 32 kHz begrenzt, da bei noch geringeren Abtastraten nicht alle wahrnehmbaren Signale fehlerfrei rekonstruiert werden können.

Abseits technischer Einschränkungen und Qualitätsanforderungen haben wir in Abschnitt 5.3.3 festgestellt, dass für die Minimierung der Pufferverzögerungen in der Audiokarte mit möglichst kleinen Blockgrößen gearbeitet werden muss. Dem Datenübertragungsschema zwischen Hardware und Betriebssystem entsprechend werden diese Größen nach unten derart beschränkt, dass das Betriebssystem nicht mehr als etwa 500 Interrupte pro Sekunde von der Audiohardware zu verarbeiten hat. Das führt typischerweise dazu, dass die minimale Blockgröße meist bei 128 Samples liegt. Unterstützen professionelle Systeme höhere Abtastraten, wird dieser Wert angepasst, sodass bspw. der E-MU 1616m bei 192 kHz eine minimale Blockgröße von 512 Samples zulässt.

In Tabelle 5.3 sind die Abhängigkeiten zwischen minimale Blockgröße, Abtastrate, Datenrate und der Blockverzögerung t_φ sowie der minimalen Verzögerungen in den Endsystemen von $5\,t_\varphi$ dargestellt. Wir sehen, dass als sinnvolle Abtastrate lediglich 44.1 bzw. 48 kHz bei einer Blockgröße von 128 Samples in Frage kommen und zu Verzögerungen in Client und Server von mindestens 13.3 bzw. 14.5 ms führen. Bereits bei eine Absenkung der Abtastrate auf 32 kHz verbleiben die Audiodaten über 20 ms in den Endsystemen. Mit dem verbliebenen Latenzbudget sind nur geringe Netzdistanzen überbrückbar, sodass sich diese Einstellung nur bedingt und für kleinere Netze eignet. Die aufgeführten Werte für die Abtastung mit 8 kHz, die bei der Telefonie (ISDN) verwendet wird, führen auch mit der minimalen Blockgröße von 128 Samples zu einer Blockverzögerung von 16 ms und zu einer Verweildauer der Daten in den Endsystemen von mindestens 80 ms, typische Werte wie sie bei der Internettelefonie auftreten und tolerierbar sind, die Anforderungen von NMP aber um

Tabelle 5.3: t_φ und Datenrate (16 bpS) in Abhängigkeit von Blockgrößen und Abtastfrequenzen

Abtastfrequenz [kHz]	Datenrate [kbps] Stereo	Mono	min. Blockgröße [Samples]	t_φ [ms]	$5 \cdot t_\varphi$ [ms]
192	6144	3072	512	2.66	13.3
48	1536	768	128	2.66	13.3
			256	5.33	26.6
			512	10.66	53.3
44.1	1411	705	128	2.90	14.5
			256	5.80	29.0
			512	11.61	58.0
32	1024	512	128	4.00	20.0
			256	8.00	40.0
			512	16.00	80.0
8	256	128	128	16.00	80.0
			256	32.00	160.0
			512	64.00	320.0

ein Vielfaches überschreiten. Eine Abtastung mit 192 kHz hat hinsichtlich Systemlatenz keinen Vorteil, da die Blockgrößen angepasst werden und zu der gleichen Gesamtverzögerung von 13.3 ms wie mit 48 kHz führen.

Prinzipbedingt erhöht sich mit steigender Abtastfrequenz auch die Datenrate linear. Das wirkt der weiteren Verarbeitung im NMP-System entgegen, die eine möglichst geringe Datenmenge anstrebt, bspw. um das Netz oder den Speicherplatz möglichst wenig zu belasten. Diesen unerwünschten Einfluss müssen wir aufgrund der Erstrangigkeit der Latenz in die Datenverarbeitung behandeln, die Parameter der Soundkarte also latenzoptimal wählen und die resultierende hohe Datenrate durch geeignete Verfahren (wie Audiodatenkompression oder Formatumwandlung) anschließend reduzieren. Mit den gegebenen technischen Einschränkungen und Möglichkeiten ist somit die Wahl der Standardparameter für die Audiohardware von 48 kHz und einer Blockgröße von 128 Samples optimal.

5.6.2 Audio-Schnittstellen

Nach der Abklärung sinnvoller Werte für t_φ ist sicher zu stellen, dass die in der Audio-Hardware vorgehaltenen Datenblöcke schnellst möglich mit dem NMP-Client ausgetauscht werden. Wie in Abschnitt 5.2 beschrieben, erfolgt der Zugriff von Programmen auf der Anwendungsschicht immer über den Kernel. Wir wollen an dieser Stelle zwei unterschiedliche Verfahren dafür vorstellen und deren Eignung für den Einsatz mit NMP untersuchen.

Das erste, ältere Verfahren ist für Einprozesssysteme ausgelegt und bildet praktisch den in Abbildung 5.2 dargestellten Datenpfad zwischen Hardware und Anwendungsschicht ab. Sobald die Soundkarte den Kernel über einen Interrupt zum Datenaustausch auffordert, erledigt der betreffende Treiber alle zeitkritischen Prozesse und ruft unmittelbar danach eine registrierte Routine innerhalb einer Anwendung für die sofortige Bearbeitung der Daten auf. Die Aktivierung der Rückruffunktion erfolgt dabei im Interrupt Kontext und wird nicht vom Scheduler gesteuert. In dieser, meist *CallBack* genannten Routine wird der zuletzt aufgezeichnete Audiodatenblock verarbeitet und der zum nächsten Interrupt auszugebende dem Treiber übergeben.

Da die Anwendung dieses Mechanismus den exklusiven Zugriff eines Programms auf die Audiokarte impliziert, ist er für moderne multimediale Betriebssysteme ungeeignet: damit der EMail Client beim Eintreffen neuer Nachrichten eine akustische Benachrichtigung ausgeben kann, während gleichzeitig ein MP3 Song abgespielt wird, müssen verschiedene Anwendungen gleichberechtigten Zugriff auf die Audio-Hardware bekommen. Dieses wird mit dem zweiten Verfahren realisiert, das heute von praktisch allen Betriebssystem für

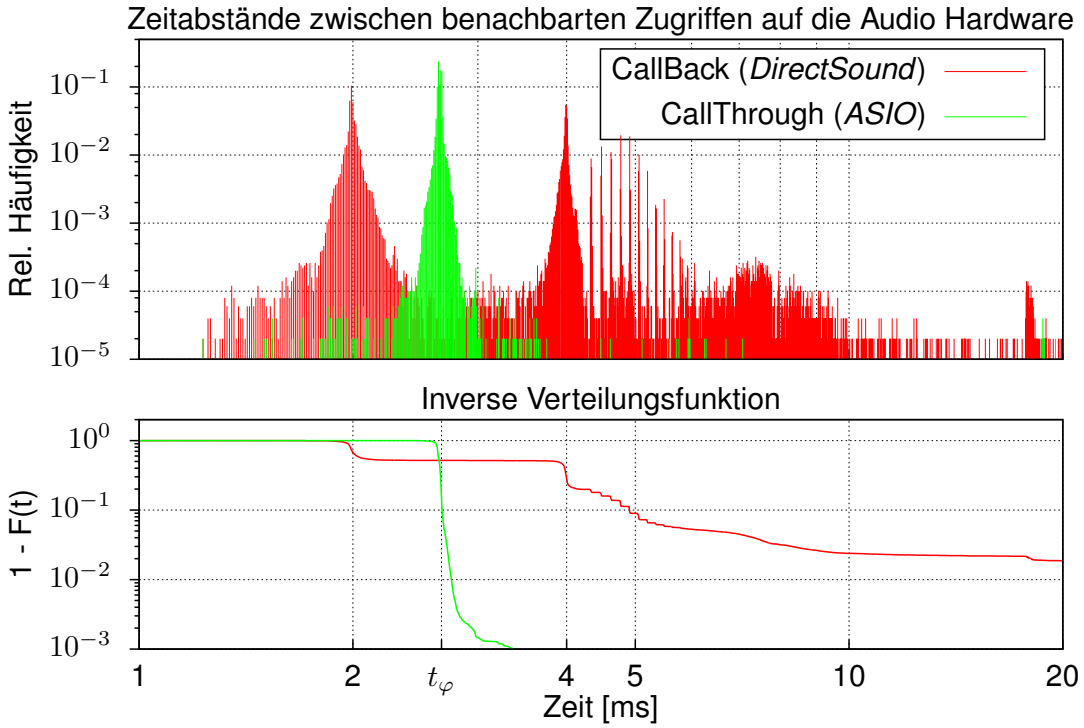

Abbildung 5.6: Latenzverhalten von *CallBack* und *CallThrough* Schnittstellen

Arbeitsplatzrechner unterstützt wird.

Kern dieses Ansatzes ist die Bereitstellung eines zentralen Audio-Mixers, der an der einen Seite exklusiv auf die Audio-Hardware zugreift und auf der anderen Seite beliebigen Anwendungen den Zugriff auf die ausgetauschten Daten ermöglicht. Jede Anwendung, die Audiodaten mit der Soundkarte austauschen will, muss dieses über den Umweg des Mixers machen. Die Anbindung erfolgt auch hierbei mit Hilfe von *CallBack* Routinen, die aufgerufen werden, sobald ein Datenaustausch erfolgen kann.

Damit unterschiedliche Zugriffszeiten und -Muster der angebundenen Anwendungen kontinuierlich synchronisiert werden können, müssen im Mixer Daten vorgehalten werden. Diese Forderung macht den zweiten Ansatz für die Verwendung mit NMP ungeeignet. Unsere Untersuchungen mit Microsofts Audioschnittstelle *DirectSound* als einen solchen Vertreter haben ergeben, dass für einen unterbrechungsfreien Betrieb mindestens acht Audioblöcke im Mixerpuffer vorzuhalten sind – mithin mehr als unser gesamtes verbliebenes Latenzbudget für NMP.

Dadurch, dass die vom Mixer verwalteten ein- und auszugebenden Datenströme vom Scheduler an die registrierten Prozesse verteilt werden, ergeben sich zusätzliche Schwierigkeiten für Echtzeitanwendungen. Zum einen verschieben sich die Ausführungszeitpunkte der betreffenden Anwendung an das Raster des Schedulers, welches mit einer zusätzlichen systematischen Verzögerung einhergeht. Zum anderen kommt die in Abschnitt 5.2 beschriebene Scheduling-Jitter Problematik zum Tragen.

Für die beiden Verfahren wollen wir eine Benennung definieren, die wir im Weiteren durchweg verwenden werden. In der Literatur werden beide Methoden als CallBack beschrieben, basierend darauf, dass die Anwendung über den Aufruf einer so definierten Routine aktiviert wird. Wir wollen den Begriff *CallBack* auch weiterhin als Bezeichnung für das zweit genannte, auf einem zentralen Audiomixer basierten, Verfahren verwenden. Das erst genannte Verfahren werden wir, mit Anlehnung an die unmittelbare Aktivierung der Anwendung über die Audio-Hardware durch den Treiber, fortan als *CallThrough* bezeichnen.

Die resultierenden Unterschiede und die zu erwartenden Probleme des *CallBack* Ansatzes bei der Iso-

Audio-Schnittstelle	Typ	Periodizität
Microsoft Windows		
DirectSound	CallBack	⊖⊖···◯
ASIO	CallThrough	⊕
Apple OS/X		
CoreAudio	CallThrough	⊕
Linux		
Jack	CallBack	⊖···⊕
ALSA	CallThrough	⊕···⊕⊕
Bewertungsskala: sehr gut ⊕⊕ … sehr schlecht ⊖⊖		

Tabelle 5.4: Periodizität der Audiozugriffe von Audio-Schnittstellen

chronität der Audioverarbeitung lassen sich mit relativ einfachen Mitteln messen. Wir haben dazu in einem NMP Aufbau über längere Zeiträume die Abstände zwischen benachbarten Aufrufen der Rückruffunktion bei Verwendung beider Methoden gemessen. Im Idealfall sollten diese jeweils einen zeitlichen Abstand von t_φ haben, jegliche Abweichung davon ist als Jitter zu werten.

In Abbildung 5.6 sind die Ergebnisse für die Messungen mit Steinbergs Audio-Schnittstelle *ASIO* als einen Vertreter des *CallThrough* Ansatzes denen von *DirectSound* gegenübergestellt. Im oberen Diagramm sind die relativen Häufigkeiten von Zeitabständen in einer logarithmischen Skala abgebildet, die untere Hälfte stellt die dazugehörige inverse Verteilungsfunktion dar. Anhand der relativen Häufigkeiten sehen wir, dass sich die Werte beim *CallThrough* sehr stark um den idealen Wert von t_φ konzentrieren, während es beim *CallBack* zwei Werte mit lokaler Konzentration bei zwei und vier Millisekunden gibt. Ganz offensichtlich arbeitet der Scheduler dieses Windows-Systems mit einer Frequenz von 500 Hz und kann die Rückruffunktion mit einer Granularität von $2\,ms$ aktivieren. Drei benachbarte Aufrufe über die Zeit von $3 * t_\varphi = 8\,ms$ setzen sich im Idealfall somit aus $\{4\,ms; 2\,ms; 2\,ms\}$-Tupel zusammen.

Weiterhin ist im oberen Diagramm die unterschiedliche Streuung der beiden Messungen deutlich sichtbar: während bei *ASIO* nur wenige Ausreißer die Zone um t_φ verlassen, ist der Streubereich bei *DirectSound* breiter und mit jeweils signifikantem relativen Anteil. Dieser Zusammenhang wird aus dem unteren Diagramm in aller Deutlichkeit sichtbar. Aufgetragen ist hier die inverse Verteilungsfunktion für die gemessenen relativen Anteile. Der nahezu ideale Verlauf beim *ASIO* mit einer steilen Flanke bei t_φ ist ein klarer Nachweis für sehr starke Periodizität. Im anderen Fall sehen wir dagegen die bereits festgestellten Sprünge bei zwei und vier Millisekunden, wobei allerdings noch ein Restanteil von knapp 20% verbleibt. Dieser wird nur langsam abgebaut, ersichtlich ist, dass der Anteil aller Zeitabstände größer als 20 Millisekunden mehr als 2% beträgt. Diese hohe Fluktuation macht deutlich, warum für einen störungsfreien Betrieb von *DirectSound* eine hohe Anzahl von Audioblöcken beim Mixer vorzuhalten ist. Für die Messungen bei *ASIO* kann dagegen abgelesen werden, dass nur 0.1% aller Messungen 3 ms übersteigen und sich diese Audio-Schnittstelle für die Verwendung mit NMP sehr gut eignet.

Mit der vorgestellten Messmethode haben wir die in modernen Betriebssystemen zur Anwendung kommenden Audio-Schnittstellen untersucht und deren Periodizität der Audiozugriffe als Kriterium für die Verwendung bei NMP in Tabelle 5.4 aufgelistet.

Wie oben aus den Messungen bereits herausgestellt, ist *DirectSound* nicht zu gebrauchen. Ein NMP-Client für Windows kann nur über die *ASIO* Schnittstelle sinnvoll realisiert werden, welches meist nur für externe Audio-Hardware für den semi- und professionellen Bereich zu finden ist. Ein unter Windows betriebener Laptop wird daher in den allermeisten Fällen mit seiner internen Audiokarte für die Verwendung als NMP-Client ungeeignet sein.

Für den Einsatz unter Linux sind die Audio-Schnittstellen *Jack* (JACK Audio Connection Kit) und der

Low-Level Zugriff auf die Audio-Hardware über das *ALSA* (Advanced Linux Sound Architecture) verbreitet. *Jack* verfolgt dabei ähnliche Ziele wie *DirectSound*, indem es Anwendungen über einen zentralen Audio-Mixer den gemeinsamen Zugriff auf vorhandene Audiogeräte bietet. Als *CallBack*-Verfahren teilt es die beschriebenen Probleme, die zu einer Erhöhung der Pufferlatenz führen. Gleichwohl kommen diese bei *Jack* nicht so stark zum Tragen, sodass mit geeigneter Hardware eine Verwendung für NMP in Frage kommt. Besser geeignet dafür ist klar *ALSA*, welches funktional das Äquivalent zu *ASIO* darstellt. Da diese Schnittstelle für alle in Linux unterstützten Audiosysteme verfügbar ist, eignet sich demgegenüber allerdings nahezu jeder beliebige Laptop ohne zusätzlicher Soundkarte als NMP-Client. Ein im Bezug auf die betrachteten Aspekte für NMP zusätzlicher Vorteil für Linux liegt in der freien Konfigurierbarkeit des Kernels und die Möglichkeit, für den Betrieb nicht erforderliche Komponenten abzuschalten. So lässt sich die Granularität und die Strategie des Scheduler an den Bedürfnissen einer Echtzeitanwendung anpassen. Darüber hinaus können nicht benötigte Treiber deaktiviert und damit die Antwortzeiten der Audio-Interrupte reduziert werden, auch ist ein vollständiger Betrieb ohne grafische Benutzeroberfläche möglich. Der in der Tabelle aufgeführte Bestwert für *ALSA* ist unter Einbeziehung der aufgeführten Maßnahmen zu verstehen, ohne diese ergibt sich eine dem *ASIO* vergleichbare Güte.

Die auf Apple-Rechnern unter dem Betriebssystem OS/X verwendete Audio-Schnittstelle *CoreAudio* ist hinsichtlich Periodizität ebenfalls mit *ASIO* vergleichbar. Gleichzeitig hat die verbaute Audio-Hardware – angesichts der Zielgruppe Musiker – meist sehr gute Qualität. Apple-Computer eignen sich daher generell sehr gut als NMP-Clients.

5.6.3 Latenzoptimales Scheduling

Neben der konstanten Verzögerung durch Pufferung entstehen in den Endsystemen an verschiedenen Stellen variable Latenzen durch die Verarbeitung auf der Anwender- und in der Kernelebene. Unsere Latenzanalyse beruht auf der Annahme, dass die Zeit für die gesamte Verarbeitung in Client und Server eine vorgegebene Grenze nicht überschreitet. Zur Abschätzung der unteren Latenzschranke in den Endsystemen haben wir in Abschnitt 5.3.4 gefordert, dass $(t_{\mathrm{SPP}} + t_{\mathrm{CPP}}) \leq \frac{t_\varphi}{n_C}$ gilt. Das entspricht bspw. in einer Sitzung mit fünf Clients einer Begrenzung der maximalen Verarbeitungszeit in Client und Server auf knapp $500\mu s$ pro Audiopaket. Zum heutigen Stand der Technik stellt dieser Wert keine besondere Hürde dar, da alle durchzuführenden Vorgänge deterministisch ablaufen und vorhersagbare Zeiten beanspruchen, die um Größenordnungen unter diesem Wert liegen.

Allerdings ist die im Zuge der Idealisierung angenommene vollständige Umsetzung der Endsysteme im Kernel aufgrund der hardwarespezifischen Abhängigkeiten nicht praktikabel. Stattdessen müssen die Anwendungen mit den vom jeweilig verwendeten Betriebssystem bereitgestellten Schnittstellen und Mechanismen so realisiert werden, dass die ermittelten Idealwerte auch praktisch nicht signifikant überschritten werden. Als zentrale Komponente, die die Analyse verwässert, haben wir die durch das Scheduling entstehende Verzögerung t_{SCHED} herausgestellt. Sie kann bei der dynamische Zuteilung von Betriebsmitteln zu sehr stark schwankenden Verzögerungen führen und damit den reibungslosen Ablauf der Anwendungen verhindern. Daneben beschränkt das zeitschlitzbasierte Zuteilen der Rechenzeit an konkurrierende Prozesse den zeitgesteuerten Ablauf auf die Größe des verwendeten Quantums. Beträgt diese bspw. 10 ms, ist es prinzipiell nicht möglich, Arbeitsabläufe mit einer Genauigkeit von 2.66 ms aktiv durchzuführen.

Dieser Problematik begegnen wir mit der Realisierung der Endsysteme als passive Prozesse, die von Ereignissen in der Hardware gesteuert und so vom Scheduler höher priorisiert werden. Im Folgenden betrachten wir die in Client und Server enthaltenen Abläufe und bestimmen analytisch, in welcher Reihenfolge diese latenzoptimal abgearbeitet werden müssen (*'gescheduled'* werden). Vorab gehen wir kurz auf das heute all-

gemein verwendete Paradigma des Multithreadings ein und erklären, warum dieses für den Einsatz in NMP ungeeignet ist.

5.6.3.1 Moderne Programmierung mit Multithreading

Threads sind quasi-gleichzeitig ablaufende Ausführungsstränge innerhalb eines Prozesses. Sie werden selbst als leichtgewichtige Prozesse bezeichnet und unterscheiden sich von diesen dadurch, dass sie im gemeinsamen Adressraum existieren.

Heute in Zeiten, in denen die Prozessorhersteller dazu übergehen, nur noch Multi-Core-Prozessoren mit mehrfach parallelen Thread-Ausführungseinheiten anzubieten, ist Multithread Programmierung naheliegend. Das etablierte Multiprocessing hat bereits vorher dafür gesorgt, dass das Betriebssystem in Multi-Core Systemen Prozesse auf alle vorhandenen CPUs verteilt. Bei der parallelen Ausführung vieler Prozesse kann so eine hohe Auslastung des Systems erreicht werden. Muss dagegen ein nicht parallelisierter und sehr CPU lastiger Prozess abgearbeitet werden, können zusätzliche CPUs nicht sinnvoll verwendet werden - die Abarbeitungsgeschwindigkeit ist auf einer Einprozessormaschine genauso hoch wie auf einem Mehrkernsystem. Hier setzt das Multi-Threading Paradigma an und gibt dem Anwender die Möglichkeit (und die Pflicht), seine Programme in Threads zu parallelisieren und so von der voranschreitenden Multi-Core Verbreitung zu profitieren. Dabei gibt es Anwendungen, die sich sehr gut in multiple Threads parallelisieren lassen (wie bspw. das zwischen Bildern unabhängige Rendern von 3D Animationsfilmen) und so sehr gut mit der CPU-Anzahl skalieren, während andere mit sequenzieller Datenabhängigkeit kaum von zusätzlichen Prozessorkernen profitieren (wie bspw. die Videocodierung mit Inter-Frame Abhängigkeiten).

Dieses Thema ist heute einer der aktuellsten Forschungsschwerpunkte, da die technische Entwicklung bei Prozessoren an einem Punkt angelangt ist, an dem die Leistung einer einzelnen CPU nicht mehr wesentlich gesteigert werden kann und stattdessen auf massive Parallelisierung gesetzt wird. Innerhalb weniger Jahre sind so die großen Hersteller wie AMD und Intel von Einzelkern-CPUs auf Multi-Cores mit im Mittel vier Kernen und mehreren Hyper-Threads pro Kern ausgewichen. Der Softwareseite wird zur Leistungssteigerung nichts anderes übrig bleiben, als über immer besseren Verfahren zur Parallelisierung der Anwendungen von Multi-Cores zu profitieren.

Wie diese Evolution der Softwareentwicklung momentan voranschreitet, lässt sich an einem aktuellen Beispiel sehr gut verdeutlichen: als Ende 2006 von Sony die neue Spielkonsole Playstation 3 vorgestellt wurde, gab es ein Aufschrei bei Technikbegeisterten und Spieleentwicklern gleichermaßen. Der als *Numbercruncher* verwendete, von IBM, Toshiba und Sony gemeinsam entwickelte *Cell* Prozessor beeindruckte allein anhand der technischen Daten. Er vereint einen Power-PC Prozessor als Verwaltungseinheit zusammen mit acht parallel arbeitenden *Synergistic Processing Elements* (SPE). Die theoretischen maximalen Leistungswerte des Cell stellten bereits die noch in Entwicklung stehenden Prozessorgenerationen der Konkurrenten deutlich in den Schatten und lösten einen entsprechenden Hype aus. Auf der anderen Seite standen die Spieleentwickler mit der Kritik, dass althergebrachte Konzepte der Spieleentwicklung praktisch nicht so parallelisierbar seien, um von den multiplen SPEs profitieren zu können. Das bewahrheitete sich an den ersten Spielen für die neue Konsole, die faktisch nur eine SPE verwendeten und die Spiele entsprechend als wenig fortschrittlich kritisiert wurden. Kein Jahr später stieg diese Auslastung auf durchschnittlich zwei bis drei SPEs, während die später entwickelten Top-Spiele wie GT5 oder GTA5 bereits eine mittlere Auslastung von über vier SPEs erreichen.

Auch auf PC-Seite wird diesem Trend gefolgt, seit absehbar geworden ist, dass künftige Leistungssteigerung nur über die Verteilung der Aufgaben auf mehrere CPUs erreicht werden kann. In der Forschung werden Algorithmen entworfen, um Probleme hoher Komplexität mit den Mitteln der Parallelausführung statt auf Spezialhardware auf Standard-PCs zu bearbeiten. Dazu zählen Projekte wie das *Folding at home*, die die

Faltung von Proteinen nicht nur auf weltweite Rechner verteilt, sondern in jedem Rechner zusätzlich alle verfügbare Möglichkeiten (bspw. in der Grafikkarte) zur Parallelisierung heranziehen.

Abseits dieser Spezialanwendungen ist der naheliegendste Ansatz, um im Alltag von vielen Ausführungseinheiten Gebrauch zu machen, das Multi-Threading. Umgekehrt wurde Multi-Threading jedoch nicht mit diesem Ziel entworfen, sondern als Paradigma zur Vereinfachung bei der Software-Entwicklung eingeführt. Jede Anwendung besteht aus mehr oder weniger vielen Funktionseinheiten mit unterschiedlichen Abhängigkeiten untereinander. Betrachten wir beispielhaft eine Anwendung mit graphischer Benutzerführung: Benutzereingaben (Maus, Tastatur, Sprache, Peripherie) müssen kontinuierlich abgefragt, graphisch angezeigt und bei Aktivierung bestimmte Funktionen ausgeführt werden. Als Einzelthread lässt sich eine solche Anwendung nur schwer umsetzen, da die verschiedenen Funktionen in unterschiedlichen Zeitabständen ausgeführt werden müssen. Während so bspw. die graphische Ausgabe mit der Bildwiederholfrequenz aktualisiert werden muss, kann die Ausführung einer komplexen Funktion mehrere Sekunden dauern. Der Entwickler müsste bei solchen Anwendungen aktiv dafür sorgen, dass alle Funktionseinheiten innerhalb der erforderlichen Intervalle ausgeführt werden.

Diesen Aufwand spart man sich durch Multithreading, indem die Funktionseinheiten der Anwendung als quasi unabhängig voneinander laufende Threads implementiert werden, die sich über definierte Mechanismen miteinander synchronisieren. Der Thread-Scheduler, der für die Auswahl des aktiven Threads innerhalb eines Prozesses verantwortlich ist, wird entweder vom Betriebssystem (wie bei Windows) oder als Teil der POSIX Thread Bibliothek *pthreads* (bei Linux und OS/X) bereitgestellt. Er wird unmittelbar nach der Aktivierung des Prozesses durch den Task-Scheduler ausgeführt und wählt unter Einhaltung von Prioritäts- und Fairnessregeln den für dieses Quantum zu aktivierenden Thread aus. Üblicherweise werden in einem Quantum mehrere Threads aktiviert, die sich nach ihrer Abarbeitung zur Synchronisation mit den anderen Threads aktiv schlafen legen. Für diese Synchronisation werden ebenfalls Funktionen vom Betriebssystem bereitgestellt, die aktives Warten auf Ereignisse oder Signale unterstützen.

Für die Signalisierung werden Funktionen bereitgestellt, die Mechanismen zum wechselseitigen Ausschluss (Mutex, mutal exclusion) von kritischen Daten- oder Codeblöcken enthalten. Da alle Threads eines Prozesses in einem gemeinsamen Adressbereich ablaufen, können Daten gemeinsam verwendet werden. Zur Wahrung der Datenintegrität darf dieses jedoch nicht gleichzeitig erfolgen, der Zugriff auf gemeinsame Daten bzw. die Ausführung gemeinsamer Codeteile muss immer abgesichert werden. Zum Schutz dieser kritischer Sektionen werden Mutexe und Semaphore verwendet, die als atomar implementierte Zähler effektiv einen Mehrfachzugriff verhindern. Diese in Multithreading Anwendungen obligatorischen Synchronisationsmaßnahmen fügen der Verarbeitungszeit systematische und stochastische Komponenten hinzu, die sich aus den Aufruf der entsprechenden Funktionen einerseits und dem Warten auf den Thread Scheduler andererseits ergeben.

Diese bereitgestellten Mechanismen vereinfachen die Umsetzung von Multithreading derart, dass es heute praktisch keine Einzelthread Anwendungen mehr gibt. In graphisch geführten Programme werden die vom Entwickler implementierten Funktionen immer als Thread aktiviert, der neben dem Rahmenwerk aus Benutzerschnittstelle und Kontrollinstanz läuft. Modernere Programmiersprachen wie Java gehen das Multithread Paradigma weiter und implementieren einzelne Funktionsaufrufe als Nebenläufigkeit, sodass ausgewachsene Anwendungen oft aus Hunderten Threads bestehen.

So komfortabel und unkompliziert sich Multithreading für den Entwickler auch darstellt – für Echtzeitanwendungen wie NMP bringt es gewichtige Nachteile mit:

Scheduling-Overhead

Damit sind die Einbußen durch den Verwaltungsaufwand für den Thread Scheduler gemeint, die bei exzessiven Threadwechseln zu Beeinträchtigungen der Effizienz führen. Dieser immanente Nachteil

relativiert sich mit steigender CPU-Leistung und ist in heutigen Arbeitsplatzanwendungen praktisch nicht mehr relevant. In zeitkritischen Applikationen wie unserem NMP, bei denen Funktionsblöcke mit Sub-Millisekunden Genauigkeiten abzuarbeiten sind, können Threadwechsel die Zuverlässigkeit beeinträchtigen.

Unvorhersagbarkeit des Thread-Schedulings

Die Selektion des aktiven Threads in jedem Zeitschlitz erfolgt dynamisch und ist daher praktisch nicht vorhersehbar. Die vom Scheduling-Algorithmus verwendeten Verfahren zur Einhaltung von Fairness und Berücksichtigung von Prioritäten unterscheiden sich zwischen Betriebssystemen und sind meist nicht dokumentiert. Dadurch wird zwar dem Anwender die Möglichkeit gegeben, die Ablaufstränge unterschiedlich zu priorisieren, eine direkte Kontrolle, in welcher Reihenfolge diese durchlaufen werden erhält er dadurch jedoch nicht.

Diese beiden Unzulänglichkeiten sind in der ersten Implementierung unseres NMP-Systems zum Tragen gekommen und haben das Verhalten negativ beeinflusst. Der erste Entwurf als Multithreading Anwendung war darauf ausgelegt, eine gewünschte Abarbeitungsreihenfolge der beteiligten Funktionsblöcke über unterschiedliche Priorisierung der betreffenden Threads zu erreichen.

In der Praxis ist das tatsächliche Verhalten der Anwendung vom beabsichtigen Verlauf mitunter stark abgewichen, wodurch bspw. die Audioschnittstelle nicht rechtzeitig bedient wurde, wenn die Abarbeitung exzessiver Benutzereingaben das gesamte Quantum erfordern. Diese Erkenntnis hat zu einem vollständigen Redesign unserer NMP-Software als Einzelthreadapplikationen geführt, in denen wir die vollständige Kontrolle über das Scheduling behalten und damit einen sehr konsistenten und zeitgenauen Programmablauf sicherstellen.

5.6.3.2 Scheduling beim Client

In Abschnitt 5.3.1 wurde der groben Ablauf im Client vorgestellt, der hier etwas genauer betrachtet werden soll. Die wesentliche Funktion ist, wie dort beschrieben, die Weiterleitung von Audioblöcken von der Audio- zur Netzwerkschnittstelle sowie in Rückrichtung, in der die empfangenen Netzpakete als Audioblöcke wieder in die Audioschnittstelle geschrieben werden. Sende- und Empfangspfad unterscheiden sich vom Prinzip und bedürfen für den Entwurf des Schedulings einer genaueren Untersuchung.

In Senderichtung gibt es für die aus der Soundkarte eingelesenen Daten keine zwingend vorgesehenen Verarbeitungsschritte, sie können unmittelbar nach dem Auslesen versendet werden. Tatsächlich haben wir dieses in unserem initialen, zum Nachweis der unteren Latenzschranke als Machbarkeitsstudie realisierten Prototypen, genau so gemacht. Für ein unter Realbedingungen praktisch einsetzbaren Client müssen dagegen in diesem Pfad Funktionen eingefügt werden, bspw. eine Audiocodierung zur Verringerung der Netzlast, Maßnahmen für Fehlerkorrekturen oder Manipulation der Audiodaten gemäß Benutzervorgaben.

In Empfangsrichtung ist der Datenpfad zwischen Netzwerk- und Audioschnittstelle durch einen De-Jitter Puffer unterbrochen. Wir habe in der obigen Analyse festgestellt, dass wegen der Asynchronität der Clients untereinander das Vorhalten mindestens eines Audioblocks vor dem Abspielen erforderlich ist, um eine kontinuierliche und lückenlose Wiedergabe zu gewährleisten. Empfangene Datenpakete werden daher zunächst in den Puffer abgelegt und erst mit Verzögerung von dort in die Audioschnittstelle geschrieben. Analog zur Senderichtung gehören auch auf diesem Pfad für die Praxis erforderliche Komponenten wie Audiodecodierung oder Behandlung von fehlenden Paketen.

Neben diesen Datenpfaden muss der Client auch eine Benutzerschnittstelle bereitstellen, um Systemparameter im Betrieb justieren zu können und über den Zustand informieren zu können. Im Gegensatz zum

Transport der Audiodaten erfordert diese Funktionalität jedoch keine zeitkritische Bearbeitung und kann so eingeplant werden, dass sie die zeitkritischen Funktionen nicht behindert.

Wir haben in Abschnitt 5.6.2 ermittelt, dass ein zeitnaher und wenig variierender Zugriff auf die Audioschnittstelle nur über *CallThrough* erreicht werden kann, wenn die entsprechenden Funktionen also direkt aus der ISR des Gerätetreibers ausgeführt wird, sobald ein Datenaustausch zwischen Applikation und Audiohardware möglich ist.

In diesem Aufruf müssen in Aufnahmerichtung aus dem Backpuffer der Audiohardware der zuletzt aufgezeichnete Audiopuffer in den User-Space kopiert und in Ausgaberichtung der als nächstes auszugebende Audioblock in die Hardware geschrieben werden. Als Minimalanforderung darf dieser Vorgang nicht länger dauern als die Zykluszeit t_φ, im Allgemeinen ist die maximal verfügbare Zeit bis zum Rücksprung aus der Routine deutlich kürzer, weil der Gerätetreiber selbst noch Operationen durchführt oder das Design der Hardware eine schnellere Bearbeitung der Anfrage erwartet. Wir müssen daher unsere Rückruffunktion so entwerfen, dass die Verweilzeit einen maximalen Wert nicht überschreitet.

Zur Ermittlung eines zuverlässigen Richtwertes haben wir Messungen mit unterschiedlichen Verweilzeiten in der Routine vorgenommen und die maximalen Werte ermittelt, bei denen ein unterbrechungsfreier Betrieb gewährleistet wird. Diese Untersuchung haben wir mit verschiedenen Soundkarten durchgeführt und ermittelt, dass bei keiner Störungen auftreten, sobald die Call-Through Routine in weniger als $0.7 \times t_\varphi$ verlassen wird. Für den Entwurf unseres Clients haben wir diesem Wert eine Reserve hinzugefügt und 1 ms (entsprechend $\frac{3t_\varphi}{8}$) als maximal zulässigen Wert definiert. Damit belastet diese Routine im Extremfall das System mit 37.5% und erlaubt im Umkehrschluss den Betrieb des zeitkritischen Teils unseres NMP-Clients auch in einem mittelmäßig ausgelastetem System.

Betrachten wir nun, was wir in dieser Millisekunde unterbringen müssen und können. Die einzige Funktion, die in der *CallThrough* Routine obligatorisch ist, ist der Austausch der Audioblöcke. Der im De-Jitter Puffer vorgehaltene als nächstes abzuspielende Block wird in die Hardware kopiert und der zuletzt aufgezeichnete Block von dort in den lokalen Speicher gelesen. In typischen Anwendungen ist die Rückruffunktion hier beendet, die Verarbeitung der Audiodaten erfolgt, sobald der entsprechende Thread der Applikation aktiviert wird. Das kann bei einem wenig ausgelasteten System unmittelbar danach erfolgen, der Scheduler kann sich jedoch auch zunächst für andere Prozesse bzw. Threads entscheiden und die Bearbeitung für mehrere Quantum-Einheiten verzögern.

In unserem Client können wir uns diese Unsicherheit nicht leisten und wollen alle zeitkritischen Abläufe in dieser Routine unterbringen. Zunächst erscheint naheliegend, die von der Audiohardware eingelesenen Audiodaten direkt zu versenden. Dieses erfolgt ohne nennenswerte Verzögerungen, da das Datenpaket lediglich in die Netzwerkschnittstelle geschrieben werden muss und nicht das erfolgte Versenden abgewartet wird.

Auf dem Sendepfad verbleiben dann die Datenmanipulation und die Audiocodierung. Für erstere sind nur lineare Funktionen wie die Pegelanpassung des Eingangssignals oder die Konvertierung des Audiodatenformates vorgesehen, allesamt Funktionen mit geringer Komplexität und hoher Datenlokalität, die meist eine vollständige Bearbeitung im Prozessorcache erlauben und damit nur sehr geringe Rechenzeiten erfordern.

Die Komplexität moderner Audiocodecs verbleibt somit als letzter kritischer Punkt auf diesem Pfad. Wir werden in Abschnitt 6.4 feststellen, dass hoch effiziente und meist hoch komplexe Codecs wegen ihrer hohen algorithmischen Verzögerungen für den Einsatz ohnehin nicht in Frage kommen. Dagegen erreichen die bei NMP verwendeten Codecs auf heutiger Rechnerhardware Bearbeitungsgeschwindigkeiten im Bereich vom 30-50 fachen der Echtzeit – die Bearbeitung eines 2.66 ms langen Audioblocks erfordert damit zwischen 50 und 90μs. Damit summiert sich die gesamte Verarbeitungszeit pro Audiopaket in Senderichtung auf unter 200 μs. Diesen Wert haben wir praktisch für die Zeitspanne zwischen dem Einlesen der Daten aus der Audio-

hardware und dem erfolgten Schreiben in die Netzwerkschnittstelle inklusive Pegelanpassung, Konvertierung von Stereo auf Mono und Audiocodierung ermittelt.

Der mit Messungen nachweisbare Erfolg dieses Ansatzes ist, dass die Audiopakete den Client kontinuierlich und mit sehr geringer Varianz verlassen. Wir können so die hoch genaue Periodizität, die die Audiohardware bietet, beibehalten und fügen in Senderichtung praktisch keinen Applikations-Jitter hinzu. Es gibt für diese Zwecke keine genauere Methode als diese, kontinuierlich äquidistante Pakete ins Netz zu senden. Daher verwenden wir immer dann, wenn wir ein NMP-typisches Paketmuster mit $t_\varphi = 2.66\,ms$ generieren wollen (wie bspw. für Laufzeitmessungen im Netz), dieses über die ISR der Audiohardware angestoßene passive Verfahren.

Von den im Client enthaltenen zeitkritischen Komponenten wurden soweit der Sendepfad komplett und vom Empfangspfad die Strecke zwischen De-Jitter Puffer und Audiohardware berücksichtigt. Die verbliebene Teilstrecke ist die zwischen Netzwerkschnittstelle und dem De-Jitter Puffer. Die Trennung ist zunächst plausibel, weil diese Komponente nicht wie die Übrigen synchron zum ISR der Soundkarte arbeitet. Der Zeitpunkt, an dem der Server die individuellen Audiopakete der Clients mischt und die gemischten Audiodaten zurücksendet, ist von den Abläufen in den Clients entkoppelt. Durch den Jitter im Netz verlieren die Empfangszeitpunkte sogar ihre Periodizität – es werden zwar im Mittel nach wie vor 375 Pakete pro Sekunde empfangen, aber nicht notwendigerweise in einem Abstand von 2.66 ms.

Da die Synchronisation mit dem ISR der Soundkarte über den De-Jitter Puffer erfolgt, besteht der intuitive Ansatz für diese Komponente darin, einen zusätzlichen Thread laufen zu lassen, der die empfangenen Pakete kontinuierlich aus der Netzwerkschnittstelle in den De-Jitter Puffer kopiert. Dieser liest blockierend und wird daher unmittelbar aktiviert, sobald ein vollständiges Paket verfügbar ist. Der Ansatz garantiert die geringste Verzögerung beim Zugriff auf die empfangenen Daten und böte sich für unser NMP an. In diesem Fall ist jedoch der hinsichtlich Latenz optimale Ansatz nicht der beste. Die oben beschriebenen Mechanismen zur Synchronisation von Threads müssen auch hierbei berücksichtigt werden, insbesondere darf kein gleichzeitiger Zugriff auf den De-Jitter Puffer erfolgen. Für diese Absicherung muss die Kontrolle an das Betriebssystem übergeben werden, was wir vermeiden wollen.

Auf der anderen Seite stellt sich bei genauer Betrachtung heraus, dass das asynchrone und der ISR vorauseilende Auslesen der Daten aus der Netzwerkschnittstelle keine praktischen Vorteile bietet. Ob ein Paket nämlich schon unmittelbar nach dem Eintreffen in den De-Jitter Puffer geschrieben wurde, oder ob dies erst in der *CallThrough* Routine erfolgt, ist für den Ablauf unerheblich. Statt also die Netzwerkschnittstelle kontinuierlich blockierend auszulesen, ist es völlig ausreichend, die empfangenen Pakete einmalig alle 2.66 ms auszulesen. Auch hierbei gilt, dass dabei im Mittel in jedem Aufruf der Rückruffunktion ein Paket auszulesen ist – wegen der Asynchronität gibt es aber auch Zyklen, in denen kein Paket auszulesen ist und Zyklen, in denen mehrere Pakete burstartig eintreffen. Der Lesezugriff auf die Netzwerkschnittstelle muss daher nicht blockierend erfolgen und alle zum jeweiligen Zeitpunkt vorhandenen Daten auslesen.

Wie für den schreibenden Zugriff gilt auch hierbei, dass das Kopieren der Pakete aus dem Empfangspuffer der Netzwerkschnittstelle in den De-Jitter Puffer sehr schnell abläuft. Mit entsprechenden Messungen haben wir ermittelt, dass hierbei auch das Auslesen von 100 Paketen innerhalb eines Zyklus die verfügbare Verweilzeit in der *CallThrough* Routine nicht übersteigt. Dieses ist eher eine theoretische Absicherung, denn sollten in einem Zyklus tatsächlich so viele Pakete zum Auslesen verfügbar sein, ist es bereits zu massiven Störungen gekommen, weil dieser Client vorher keine Daten zum Abspielen hatte.

Der maximale Wert von Paketen, die innerhalb eines Zyklus ausgelesen werden können, ohne dass es zu Störungen bei der Audiowiedergabe kommt, kann analytisch einfach ermittelt werden. Aufgrund der Granularität der Latenz betrachten wir als maximal tolerierbare Anzahl verzögerter Pakete $n_{max} = \lfloor \frac{30\,ms}{t_\varphi} \rfloor = 11$. In der Soundhardware muss somit der gerade ausgespielte Audioblock vor spätestens elf Zyklen aufgezeichnet

worden sein. Da sich laut obiger Analyse die Verzögerungen in Hardware und Gerätetreiber auf drei Blöcke summieren, darf die Länge unseres Synchronisations- und De-Jitter Puffers acht Pakete nicht überschreiten, wenn eine Systemverzögerung von 30 ms nicht überschritten werden soll. Werden in einem Zyklus mehr Pakete aus der Netzwerkschnittstelle gelesen, können die ältesten Pakete im Zuge der Resynchronisation nach der aufgetretenen Störung verworfen werden. Auf die Problematik des Pufferhandlings kommen wir genauer in Abschnitt 6.5 zu sprechen, das Thema Synchronisation wird in den Abschnitten 6.6 und 6.7 genauer untersucht.

Orientieren wir uns bei der Umsetzung des Clients an das in Abbildung 5.2 dargestellte Schema, können Bursts von eintreffenden Paketen innerhalb eines Zyklus Probleme verursachen. Im skizzierten Ablauf werden die in der Netzwerkschnittstelle bereitgehaltenen Pakete ausgelesen, verarbeitet und in den De-Jitter Puffer eingereiht. Im Falle eines burstartigen Paketankuftmusters sind dabei in manchen Zyklen keine neu empfangenen Pakete vorhanden, während in anderen multiplikativer Aufwand für die Bearbeitung und die Decodierung mehrerer Pakete anfällt. Das widerspricht unserem Ziel, eine kontinuierliche und gleichmäßige Bearbeitung des isochronen Datenstroms zu gewährleisten. Ein naheliegender Lösungsansatz für dieses Problems liegt darin, in jedem Zyklus genau ein Paket aus dem Empfangspuffer zu lesen und zu verarbeiten. In einem idealen System mit konstanter Verzögerung im Netzwerk und in den Endsystem führt dieser Ansatz auch zum gewünschten Ergebnis.

In der Praxis würde dagegen der existierende Jitter immer dazu führen, dass jedes in einem Zyklus nicht rechtzeitig bearbeitete Paket einen kontinuierlichen Anstieg der im Empfangspuffer vorgehaltenen Pakete führt und die Latenz erhöht. Das frühestmögliche Leeren des Empfangspuffers ist daher unvermeidbar, unabhängig davon, wie viele Pakete in einem Zyklus fallen. Diese Forderung ist praktisch aus zwei Gründen nicht hinderlich. Zum einen haben wir festgestellt, dass das alleinige Einlesen der Daten aus dem Empfangspuffer auch bei einer großen Anzahl von Paketen zu keinen nennenswerten Latenzen führt. Andererseits beschränkt die Länge des De-Jitter Puffers die Anzahl der innerhalb eines Zyklus gültigen Pakete für eine störungsfreie Ausgabe. Das Auslesen der Pakete aus dem Empfangspuffer ist somit entweder für die Latenz nicht relevant oder impliziert bereits eingetretene Probleme, die ohnehin zu einer Störung der Audiowiedergabe geführt haben.

Anders verhält es sich mit den Zeiten für die Verarbeitung und Decodierung der empfangenen Audiopakete, denn diese sind mit teilweise hoher algorithmischer Komplexität behaftet und verursachen eine merkliche Verzögerung durch CPU-Zeit. Zwar ist auch hierbei die Anzahl möglicher Pakete innerhalb eines Zyklus bei störungsfreier Wiedergabe nach oben beschränkt, allerdings kann eine Vielfache Ausführungszeit t_{OP} bei leichtgewichtigen Clients bereits zu einer Überschreitung der Zykluszeit führen und eine störungsfreie Verarbeitung behindern. Diese Gefahr vermeiden wir dadurch, dass wir ankommende Pakete zunächst unverarbeitet in den De-Jitter Puffer einreihen und die pro Paket durchzuführenden Operationen erst unmittelbar vor dem Kopieren der Audiodaten in den Abspielpuffer der Soundkarte erfolgen. Damit wird sicher gestellt, dass in jedem Zyklus genau ein Audiopaket bearbeitet wird und Pakete, die aufgrund ihres verspäteten Empfanges nicht mehr ausgegeben werden können, keine unnötige CPU-Zeit verbrauchen.

Diese Maßnahmen gewährleisten eine weitest gehende isochrone Bearbeitung der Audiodaten und entspricht damit der Kernanforderung an den NMP-Client. Für die Praxis ist daneben die Schnittstelle zum Benutzer nötig und erfordert in diesem Zusammenhang eine dedizierte Analyse. Wir haben im vorigen Abschnitt dargestellt, dass eine Interaktion mit dem Benutzer praktisch immer über das Multithreading implementiert wird. Typischerweise läuft dabei der GUI-Thread mit einer relativ geringen Priorität nebenher und kümmert sich darum, dass Benutzereingaben verzögerungsarm ausgewertet und die Bildschirmausgabe zeitnah aktualisiert wird. Für die Geschwindigkeit, mit der das visuelle Feedback erfolgen muss, gibt es die technische obere Schranke der Bildwiederholfrequenz. Es macht demnach keinen Sinn, Ein- und Ausgabe

häufiger zu aktualisieren, als es vom Benutzer wahrnehmbar ist. Blicken wir zur Verdeutlichung bspw. auf die Computerspiele als Anwendungen mit den höchsten Anforderungen an visueller Reaktivität: der Spielablauf wird ab einer Bildwiederholrate von 30 Bildern pro Sekunde (fps) als flüssig wahrgenommen. Um wahrnehmbares Flimmern zu vermeiden, liegen typische Bildwiederholfrequenzen von Monitoren meist bei 60 fps. Diese technische Vorgabe legt fest, dass eine Aktualisierung der visuellen Ausgabe in kürzeren Intervallen keinen Sinn macht. Die kleinste sinnvolle Zykluszeit liegt somit bei 16.6 ms.

Für den NMP-Client benötigen wir eine zeitunkritische Instanz, die mit dieser Häufigkeit in den Audiodatenfluss eingreift und den Ablauf parametrisiert bzw. Zustände abfragt. Aus den im letzten Abschnitt beschriebenen Einschränkungen bei der Synchronisation von Threads folgt, dass Multithreading für den Client nicht verwendbar ist. Innerhalb der ISR, aus deren Kontext heraus unsere *CallThrough* Routine ausgeführt wird, darf auf keinen Fall aktiv gewartet werden, insbesondere können keine Funktionen zum exklusiven Zugriff auf Daten ausgeführt werden. Als Konsequenz muss bspw. die Benutzerschnittstelle als separater Prozess laufen, der über entkoppelnde IPC-Mechanismen an diese Komponente angebunden ist.

Wir werden bei der Diskussion der Implementierung unseres NMP-Systems in Kapitel 7 ausführlich die realisierte Trennung des Echtzeit-Clients *RC*, der Inhalt dieser Analyse ist, vom zeitunkritischen Client *NC* eingehen. An dieser Stelle beschränken wir uns auf die Festlegung der Bedingungen für die Interprozesskommunikation (IPC) zwischen diesen beiden Komponenten. Diese bedarf einer genaueren Betrachtung, weil die Kommunikation zwischen zwei Modulen mit höchst unterschiedlichen Verhalten stattfindet: der RC arbeitet synchron und darf durch den asynchronen Datenaustausch mit dem zeitunkritischen Teil nicht in seiner Periodizität behindert werden. Auf technischer Ebene bedeutet dies, dass die Kommunikation der Zykluszeit untergeordnet wird.

Das für die Kommunikation zwischen den beiden Komponenten implementierte Protokoll wird im Kapitel 8 beschrieben, auf die Eckdaten greifen wir hier zur Verdeutlichung etwas vor. Zu beachten ist, dass der NC nicht nur die eingangs anvisierte Benutzerschnittstelle implementiert – vielmehr werden alle Funktionen, die nicht unmittelbar zum Audiodatentransport zwischen Netz- und Audioschnittstelle gehören, dorthin ausgelagert. So gehören die Programmteile zum lokalen Archivieren der Audiodaten, die Verarbeitung und Darstellung des Systemstatus oder die Auswertung statistischer Daten ebenso dazu. Zum Austausch dieser Informationen wird eine IPC über die Socket Schnittstelle verwendet. Das Protokoll ist in Richtung vom NC zum RC als Anfrage-Antwort System ausgeführt, in der Gegenrichtung werden nur unidirektionale Report- bzw. Statusnachrichten versendet. Über diese berichtet der RC periodisch über seinen Zustand (um bspw. ein Level-Meter darstellen zu können) oder versendet Kopien von Audiopaketen zur temporären Archivierung. Die Anfragen vom NC kommen dagegen asynchron herein und werden mit einer Antwort quittiert.

Für das angestrebte Ablaufverhalten sind die periodischen Nachrichten des RC aufgrund ihres vorhersehbaren Auftretens unproblematisch. Dagegen können asynchron eintreffende Anforderungen zu Störungen bei der Verarbeitung führen, wenn deren Bearbeitung nicht explizit geplant und hinsichtlich der Einhaltung der verfügbaren Zykluszeit ausgeführt wird. Dabei müssen Vorkehrungen getroffen werden, die eine Überlastung des RC durch eine Überhäufung von Anfragen vermeiden. Intuitiv scheint diese Problematik derer beim burstartigen Empfang von Audiopaketen zu ähneln: auch dabei musste verhindert werden, dass die Anzahl von zu bearbeitenden Audiopaketen innerhalb eines Zyklus variiert. Analog bietet sich an, eintreffende Anfragen zunächst in einem De-Jitter Puffer einzureihen und in jedem Zyklus eine maximale Anzahl dieser Anfragen zu bearbeiten. Einen solchen Mechanismus haben wir prototypisch implementiert und damit in synthetischen Tests nachgewiesen, dass auch höchst asynchrone und in Häufungen auftretende Anfragen abgearbeitet werden können, ohne die zyklische Konstanz der Verarbeitung zu beeinträchtigen.

Im weiteren Verlauf haben wir diesen Ansatz durch ein einfaches und robustes Verfahren abgelöst. Es basiert darauf, dass die Häufigkeit potentieller asynchroner Benutzeranfragen beschränkt ist – ein Anwender

kann physiologisch nur eine limitierte Anzahl von Interaktionen pro Zeiteinheit durchführen: selbst ein unkontrolliertes Betätigen der Tastatur kann nicht mehr als etwa 20 Ereignisse pro Sekunde generieren. Auch zusammen mit periodischen Statusanfragen für die Aktualisierung des visuellen Feedbacks, die höchstens mit der Bildwiederholfrequenz erfolgen müssen, ist die Anzahl potentieller Anfragen pro Sekunde deutlich kleiner als die Anzahl von 375 Zyklen innerhalb dieser Zeit. Der nach dieser Betrachtung realisierte Ansatz ist so einfach wie robust: der NC versendet jeweils nur eine Anfrage und wartet dann auf die Antwort des RC, die im nächsten Zyklus und daher im Mittel nach 1.33 ms generiert wird. Die Antwortzeit unterscheidet sich nicht vom ersten Ansatz, bei dem die Anfragen zunächst gepuffert und kontinuierlich verarbeitet werden.

Mit dieser Überlegung können wir den RC als *CallThrough* Routine implementieren, die aus der ISR der Audiohardware aktiviert wird und mit der größtmöglichen Genauigkeit abläuft. Auf Nebenläufigkeiten verzichten wir vollständig und vermeiden so das in seiner Ausführung variierende Scheduling durch das Betriebssystem. In Algorithmus 1 ist diese Komponente vereinfacht dargestellt.

Algorithmus 1 : Prinzip des Echtzeit-Clients RC

Input : CallThrough Aufruf innerhalb der ISR
lies Audiopaket n_i vom Audioeingabepuffer;
bearbeite Audiopaket n_i;
sende Audiopaket n_i zum Server;
while *Pakete im Empfangspuffer vorhanden* **do**
  kopiere Audiopaket vom Empfangspuffer in De-Jitter Puffer;
end while
berechne Abspielverzögerung d_p;
lese Audiopaket n_{i-d_p} aus De-Jitter Puffer;
bearbeite Audiopaket n_{i-d_p};
schreibe Audiopaket n_{i-d_p} in Audioausgabepuffer;
if *Anfrage vorhanden* **then**
  bearbeite und beantworte Anfrage;
end if
sende Statusupdate;
Result : Verweilzeit in der Call-Through Routine $t_C \leq 1\,ms < t_\varphi$

Das Arbeitsverhalten des RC ist somit deterministisch und zyklisch konstant, d.h., die für jeden Zyklus erforderliche CPU-Zeit variiert nur sehr wenig und ist nach oben begrenzt, da die in jedem Zyklus durchgeführte Verarbeitung sowohl für Audiopakete als auch für Anfragen limitiert ist. In Kapitel 7 werden wir darstellen, wie die strikte Trennung zwischen zeitkritischen und -unkritischen Komponenten zum Aufbau verschiedenster Topologien von NMP-Systemen befähigt. Dort wird auch thematisiert, wie die Sicherstellung der zyklischen Verarbeitung erlaubt, diesen Teil des Clients sehr leichtgewichtig bspw. als eingebettetes System zu realisieren.

5.6.3.3 Scheduling beim Server

Anders als mit der Audiohardware beim Client verfügt der Server über keine eigene aktive Synchronisationsquelle, die eine kontinuierliche und zyklische Aktivierung der Datenverarbeitung ermöglichen kann. Das über Zeitgeber gesteuerte periodische Aktivieren des Servers ist aus den bereits geführten Überlegungen nicht anwendbar, weil damit die Abhängigkeit vom Scheduler des Betriebssystem einhergeht, der unsere strikten Anforderungen nicht gewährleisten kann. Stattdessen muss ein alternatives Scheduling Konzept implementiert werden, welches eine verzögerungsarme Bearbeitung der empfangenen Daten sicher stellt.

Bei der Betrachtung der Datenpfade in einem NMP-System erkennen wir, dass für den Server bereits ein idealer Taktgeber in Form der eintreffenden Pakete existiert. Mit den im letzten Abschnitt beschriebenen Maßnahmen ist sicher gestellt, dass der Client Pakete periodisch und mit sehr geringer Variabilität zum Server versendet. Im idealen Netzwerk ohne Jitter wären die Paketzwischenankunftszeiten ebenso konstant periodisch, in jeder Zykluszeit t_φ käme dann genau ein Audiopaket beim Server an. Im realen Netz variiert die Paketzwischenankunftszeit mit dem Jitter im Netz, im Mittel empfängt der Server 375 Pakete pro Sekunde.

Überlegen wir uns an dieser Stelle, ob die Ereignisse eintreffender Audiopakete als alleiniger Taktgeber für die vollständige Funktionsfähigkeit des Servers ausreichend ist. Betrachten wir dafür ohne Einschränkung der Allgemeinheit eine Session mit nur einem Client: der Client sendet seine Audiopakete zum Server, dieser reiht diese nach Empfang in sein De-Jitter Puffer ein und sendet sie unmittelbar danach zum Client zurück. Die zentrale zu klärende Frage hierbei ist nun: ist es ausreichend, wenn der Server jeweils nur dann aktiv wird, wenn ein Paket empfangen wird, oder sind Situationen denkbar, in denen er autonom (bspw. Zeit gesteuert) agieren muss?

Bevor wir die Frage untersuchen, stellen wir zunächst klar, dass wir hier ausschließlich den Server in seiner Funktion als Echtzeit-Audio-Mischer betrachten. Allen über diese Grundfunktion hinausgehenden Dienste, die den Mehrwert unseres NMP-Systems definieren, widmen wir uns anschließend. Mit dieser Vorgabe kann schnell gezeigt werden, dass es keine sinnvollen Zeitpunkte gibt, in denen der Server außerhalb der Paketankunftszeitpunkte autark aktiv werden muss. Der formale Nachweis ist einfach wie einleuchtend: das Vorhandensein von Audiopaketen p mit einer bestimmten Sequenznummer i ist ein binärer Zustand, zu jedem Zeitpunkt ist ein Paket p_i entweder beim Server bereits eingetroffen oder noch nicht. Diese Zustände ändern sich nur und ausschließlich durch den Empfang eines Paketes. Sinnvolle Operationen und Datenverarbeitungen können entweder unmittelbar danach ausgeführt werden oder in diesem Zyklus gar nicht. Es macht für den Server somit keinen Sinn, zwischen zwei Paketempfangszeitpunkten aktiviert zu werden.

Intuitiv scheint es für diese Behauptung Widersprüche zu geben, von denen der naheliegendste die Erkennung und Behandlung von Paketverlusten und Stauungen ist. Betrachten wir dazu folgende Situation: auf dem Netzpfad vom Client zum Server wird eine Netzkomponente überlastet und staut die Audiopakete für $t_c = 5\,t_\varphi$. In dieser Zeit ist der Server passiv und bekommt von diesem Problem nichts mit. Stattdessen wird er erst aktiviert, wenn die gestauten Pakete – dann wahrscheinlich burstartig – eintreffen. Die Stauung kann so erst nachträglich erkannt und bearbeitet werden. Die prinzipielle Frage ist nun: könnte das Systemverhalten nicht verbessert werden, wenn der Server das Problem früher erkennt. Betrachten wir den alternativen Fall, in dem der Server Zeit gesteuert alle 2.66 ms aktiviert wird (Anm.: diese Betrachtung wäre wegen der bekannten Einschränkungen beim Scheduler zunächst eine theoretische). Nehmen wir weiterhin an, der Server toleriert einen maximalen Jitter von $2 \cdot t_\varphi$, dann würde er die Stauung im Netz bereits $3 \cdot t_\varphi$ nach Eingang des letzten Paketes vor dem Aussetzer erkennen.

Die Detektion wäre somit tatsächlich früher möglich, allerdings ist diese Information nicht sinnvoll verwertbar. Der Server könnte nach der Erkennung des Problems den Client darüber informieren und ihm eine frühzeitige Behandlung ermöglichen. Da er jedoch keine Audiodaten bereitstellen kann, stellt sich dabei die Frage, welcher Profit sich für den Client damit einstellt. Oder anders formuliert: inwieweit kann der Client von einer Nachricht vom Server profitieren, die ihm mitteilt, dass Lücken im Audiodatenstrom aufgetreten sind gegenüber der eigenen Detektion dieser Lücke? Eine solche Detektion ist in jedem Fall auch beim Client zu implementieren, um Probleme auf dem Rückpfad zu erkennen. Unsere Analysen und praktischen Untersuchungen haben diese Frage klar verneint. Es gibt keinen praktisch relevanten Fall, in dem eine Aktivierung des Servers zwischen Paketankunftszeitpunkten zu einer Verbesserung des Ablaufverhaltens unseres NMP-Systems führen würde.

Die Erweiterung der hier gemachten vereinfachten Analyse auf mehrere Clients ist intuitiv nachvoll-

ziehbar und bleibt dem Leser überlassen. Wir werden auf die Besonderheit dieser Passivität des Servers, der ausschließlich über den Empfang von Audiopaketen aktiviert wird, im weiteren Verlauf an verschiedenen Stellen zu sprechen kommen. So wird die Dimensionierung und das Handling der Puffer in Abschnitt 6.5 aufgegriffen, während die Synchronisierung ohne Zeitgeber in Abschnitt 6.7 thematisiert wird.

Abschließend wollen wir sicherstellen, dass die eingangs gestellte Beschränkung bei der Betrachtung auf die Kernfunktion des Servers als Audio-Mixer wieder gelockert werden kann und ein solcher passiver Server die erweiterten Funktionen bereitstellen kann. Unser NMP-System ist so konzipiert, dass mit den während der Echtzeit-Sitzungen gewonnenen Audiodaten zahlreiche Mehrwertdienste abrufbar sind. Diese decken bspw. den Ersatz einzelner oder mehrerer Musiker durch bereits aufgezeichnete Audiospuren ab, bis hin zu passivem Abruf ganzer Stücke. Vom technischen Anspruch her liegen diese Dienste unterhalb der Mixer-Funktionalität und erreichen bei der Implementierung eine Bandbreite zwischen der Umsetzung virtueller Clients und Bereitstellung von Streaming. Auf die einzelnen Komponenten kommen wir genauer in Kapitel 7 zu sprechen. Für den Entwurf des Schedulings ist hier relevant, dass der Server sowohl aus zeitkritischen als auch -unkritischen Komponenten besteht. Die zeitkritischen sind dabei offensichtlich alle Funktionen, die zum Audio-Mischen in Live-Sitzungen gehören, zeitunkritisch sind die Archivierung, Nachbearbeitung und das Streaming von Audiodaten, ebenso die gesamte Verwaltungsfunktionalität.

Ähnlich dem Client besteht auch beim Server durch die Überlagerung dieser zwei Typen von Funktionen die Gefahr, dass die latenzkritischen Komponenten durch zeitunkritische und langsame Funktionen gestört werden. Veranschaulichen wir uns das anhand der obligatorischen Archivierung der Audiodaten. Damit reale Musiker durch virtuelle ersetzt und die ganze Session abgerufen werden kann, müssen am Server sowohl alle Einzelspuren als auch die während einer Sitzung gemischte Spur permanent gespeichert werden. Es erscheint naheliegend, jedes empfangene und versendete Audiopaket auf Festplatte zu speichern. Das funktioniert so lange gut, bis der Schreib-Cache des Festplattentreibers voll ist und der physikalische Zugriff auf die Festplatte erfolgt. Wie am Anfang dieses Abschnitts beschrieben, ist dieser um mehrere Größenordnung langsamer als unsere Zykluszeit. Im schlechtesten Fall blockiert der Schreibaufruf den NMP-Server um ein Vielfaches von t_φ, sei es dadurch, dass die CPU den Prozess bis zum erfolgreichen Schreiben vollständig blockiert oder dadurch, dass der Cache-Dienst mehrfach gescheduled und der Server wiederholt unterbrochen wird.

Diese potentiellen Probleme sind uns schon beim Client begegnet. Daher liegt es nahe, das gleiche Verfahren anzuwenden und den Server in eine zeitkritische und eine -unkritische Komponente aufzuteilen. Wieder brechen wir die kritische Komponente auf die reine Funktionalität des Audio-Mixers herunter und lagern alles andere in den unkritischen Teil aus. Um trotz fehlender Möglichkeit einer Aktivierung des Servers aus einer ISR heraus (wie beim Client geschehen) eine zeitnahe und vom Scheduler möglichst unabhängige Bearbeitung zu erreichen, gehen wir wie folgt vor: wir führen den Echtzeit-Server *RS* als Prozess mit der höchstmöglichen Priorität aus, der nur durch den Kernel unterbrochen werden kann. Er muss zwar immer noch vom Scheduler aktiviert werden, da er jedoch aufgrund der Priorität immer an erster Stelle in der Warteschlange steht, beeinflussen die Variabilitäten des Schedulers nicht die Ausführung.

Analog zum Client wird auch hier eine Kommunikation zwischen den beiden Servertypen verwendet, die zum RS auf dem synchronen Anfrage-Antwort Konzept beruht und asynchrone Nachrichten für die Archivierung von Audiodaten und periodischer Statusaktualisierungen in Rückrichtung ermöglicht.

Der Server wird als Daemon realisiert, der blockierend auf die Netzschnittstelle zugreift und unmittelbar aktiviert wird, sobald ein Audiopaket von den Clients oder eine Anforderung vom zeitunkritischen Server *NS* vorliegt. Der prinzipielle Ablauf ist in Algorithmus 2 vereinfacht dargestellt.

Algorithmus 2 : Prinzip des Echtzeit-Servers RS

```
Input : Blockierender Zugriff auf Netz-Schnittstelle
Input : Ausführung mit höchstmöglicher Priorität
while Daten im Empfangspuffer vorhanden do
    while Audiopakete vorhanden do
        lese Audiopaket aus;
        sende Audiopakete zur Archivierung;
        if Mischen ist möglich then
            synchronisiere und erzeuge Mix-Paket;
            sende Mix-Paket an alle Clients;
            sende Mix-Paket zur Archivierung;
        end if
    end while
    if Anfrage vorhanden then
        bearbeite und beantworte Anfrage;
    end if
end while
```

5.7 Relevanz der Analyse für reale Szenarien

Mit den durchgeführten Überlegungen haben wir Richtwerte über die Verweilzeiten der Audiodaten in den Endsystemen gewonnen und mit einer Abschätzung der Verzögerungen im Netz die Größenordnung für den maximalen theoretischen Einsatzradius von NMP ermittelt. In diesem Abschnitt wollen wir beleuchten, welche Aussagekraft diese Überlegungen für den praktischen Einsatz bergen.

Bei der Latenzanalyse in den Endsystemen haben wir festgestellt, dass sich der zu erwartende Wert aus vorhersagbaren und variablen Komponenten zusammensetzt. Deterministisch sind dabei die Verzögerungen, die durch Pufferung und das reine Abarbeiten von Prozessorinstruktionen entstehen. Nicht vorhersehbar sind Verzögerungen, die durch die gemeinsame Nutzung von Ressourcen, bei der Bearbeitung asynchroner Ereignisse oder datenabhängig unterschiedlichen Durchläufen von Algorithmen entstehen.

Als Ergebnis unserer Überlegungen haben wir in Formel (5.5) die konstante Pufferverzögerung $5 \cdot t_\varphi$ als untere Schranke für die Verweilzeit der Daten in den Endsystemen ermittelt. Diese ist prinzipbedingt nicht zu vermeiden und wird in der Praxis um den in der Formel ermittelten Betrag von $(n_C + 1) \cdot \left(t_{IRQ} + t_{SCHED} + t_{IP} + t_{OP}\right)$ ergänzt, um Interrupt-Antwortzeiten, Scheduling und Datenverarbeitung zu berücksichtigen. Bereits hier können wir plausibel annehmen, dass diese zusätzlichen Verzögerungen auf vernachlässigbar kleine Werte begrenzt werden können. So kann auf eine Weiterverarbeitung der Audiodaten in Form von Codierung und Fehlerbehandlung gänzlich verzichtet werden um t_{IP} und t_{OP} zu minimieren. Die stark variierende Zeit für das Scheduling kann durch die Verwendung von Echtzeit-Betriebssystemen nach oben auf kleine Werte begrenzt werden, Gleiches gilt für die Bearbeitung von Interrupts.

Im vorangehenden Abschnitt 5.6 haben wir gezeigt, dass wir auf Seiten der Endsysteme die theoretisch bestimmte untere Schranke der Latenz mit geeigneten Mitteln auch praktisch umsetzen können. Dafür wird keine spezielle Hardware oder Echtzeitbetriebssysteme benötigt, vielmehr ist die Realisierung eines NMP-Systems mit handelsüblichen Computern unter Verwendung moderner Desktop Betriebssysteme möglich.

Leider gilt diese Übertragbarkeit der theoretischen Betrachtung auf die Praxis nicht für die Netzlatenz. Die Grundannahme, dass eine untere Schranke für die Verzögerung anhand der Netzdistanz und der Übertragungsverzögerungen in den Routern abgeschätzt werden kann, lässt sich praktisch nur sehr selten verwerten. Die Route zwischen NMP-Client und Server lässt sich oft gar nicht oder nur mit erweiterten Kenntnissen der Netztopologie abschätzen. Auch wenn diese bekannt ist, erfolgt die Wegewahl nach internen Richtlinien der

Betreiber, sodass eine Bestimmung der Netzdistanz und damit der Ausbreitungsverzögerung praktisch nicht möglich ist. Ähnliches gilt für die Übertragungs- und Verarbeitungsverzögerungen in den Routern. Der genaue Aufbau der Infrastruktur auf dem Netzpfad wird in den seltensten Fällen bekannt sein, besonders wenn die Route über kommerzielle Carrier verläuft.

Zur Verdeutlichung dieser Problematik betrachten wir die Netzverzögerung in einem Aufbau mit einem NMP-Server am IBR und je einen Client an der TU Darmstadt und der Uni Frankfurt - die beiden Clients liegen also geografisch nur wenige Kilometer auseinander. Wir können dabei auf die Information zurückgreifen, dass alle Universitäten in Deutschland über das DFN verbunden sind. Das seit 2006 verwendete X-Win Backbone betreibt Peering zu kommerziellen deutschen Carriern und ist am Europäischen Forschungsbackbone GÉANT2 angeschlossen [2]. Abbildung 5.7 zeigt die vom DFN bereitgestellte Skizze der logischen Verbindungen an. Die Anschlusskapazität reicht von 1 Gbps an den Außenbereichen bis hin zu 1 Tbps im Kernbereich.

Abbildung 5.7: X-Win Backbone

Mit Kenntnis dieser Angaben können wir annehmen, dass die Route zwischen dem IBR und den beiden Clients sehr ähnlich verlaufen wird. Sie sollte sich bis Frankfurt für beide Verbindungen gleichen und über ein kurzes Teilstück weiter nach Darmstadt führen. Dabei sollte sie in Abhängigkeit von der momentanen Auslastung mit hoher Wahrscheinlichkeit entweder über Aachen oder Leipzig laufen.

Für die Abschätzung der minimalen Laufzeit greifen wir auf Routenplaner zurück, die für die kürzeste Strecke zwischen dem NMP-Server und den Clients 390 bzw. 420 km ermitteln. Die Ausbreitungsverzögerung für die Strecke in beiden Richtungen beträgt entsprechend ungefähr 3.9 bzw. 4.2 ms.

Für den Weg vom IBR zur Uni Frankfurt wird mit *traceroute* [17] die in Abbildung 5.8 ausgegebene Route ermittelt. Wir entnehmen dem Log, dass die Route vom IBR nach vier Hops und weniger als 1 ms im X-Win Backbone (188.1.0.0/16) mündet. Über Hannover geht es innerhalb des X-Win direkt zum Frankfurter Knoten, und von dort aus zur angebundenen Universität.

Im Vergleich dazu fällt zunächst überraschend die Route zum 30 km weiter südlich gelegenen Darmstadt aus, die in Abbildung 5.9 ausgegeben ist. Auch hier führt der Weg von Braunschweig zunächst zum selben Einstiegsknoten im X-Win. Im DFN geht es weiter über Magdeburg nach Potsdam. In Berlin findet ein Peering in das Netz von TeliaSonera statt, in dem es über Umwege über Hamburg nach Frankfurt und schließlich nach Darmstadt geht.

Warum dieser vermeintliche Umweg gewählt wird, offenbart sich erst nach einem Blick auf das TeliaSo-

```
Routenverfolgung zu www.uni-frankfurt.de [141.2.22.39]
über maximal 30 Abschnitte:
  1    <1 ms  corona.ibr.cs.tu-bs.de [134.169.34.1]
  2    <1 ms  134.169.39.254
  3    <1 ms  xwin-bs-gi4-4.rz.tu-bs.de [134.169.3.45]
  4    <1 ms  188.1.46.145
  5     2 ms  xr-han1-te2-1.x-win.dfn.de [188.1.145.14]
  6     2 ms  zr-han1-te0-0-0-1.x-win.dfn.de [188.1.145.214]
  7    14 ms  zr-fra1-te0-7-0-2.x-win.dfn.de [188.1.145.126]
  8    15 ms  xr-fra1-te2-1.x-win.dfn.de [188.1.145.189]
  9    15 ms  kr-uni-frankfurt.x-win.dfn.de [188.1.42.38]
 10     *     Zeitüberschreitung der Anforderung.
 11     *     Zeitüberschreitung der Anforderung.
 12    15 ms  alberto.net.uni-frankfurt.de [141.2.252.50]
 13    15 ms  traminer.rz.uni-frankfurt.de [141.2.22.39]
```

Abbildung 5.8: traceroute zwischen dem IBR und der Uni Frankfurt

nera Backbone. Wie in Abbildung 5.10 dargestellt, unterhält der Provider eine breitbandige Kernverbindung zwischen Hamburg und Frankfurt, über die er den Verkehr von Berlin leitet. Wir können spekulieren, dass die TU Darmstadt über TeliaSonera am DFN angeschlossen wäre, die in Frankfurt kein Peering zum X-Win betreibt und deswegen der Umweg über Berlin gegangen wird. Es können auch andere Gründe wie Volumenverträge zwischen den Anbietern oder zum Zeitpunkt der Messung überlastete Router auf bestimmten Teilstrecken diesen verlängerten Netzpfad verursachen.

Damit bestätigt sich die oben aufgestellte Behauptung, wonach eine Abschätzung der Netzdistanz aufgrund geografischer Daten nicht möglich ist. Wir sehen darüber hinaus, dass auch bei Kenntnis der Netztopologie (wie hier des X-Win und des TeliaSonera Backbones) aufgrund der dynamischen Wegewahl keine Vorhersage über die Netzdistanz getroffen werden kann.

Ungeachtet dieser Abweichungen können wir feststellen, dass die Abschätzung der Netzlatenz anhand der Signalausbreitungszeit über die ermittelte Netzdistanz und einer angenommenen Verweildauer von etwa 0.5 ms je Router plausibel ist und für die hier exemplarisch durchgeführten Messungen brauchbare Näherungswerte liefert.

Als Ergebnis dieser Veranschaulichung können wir feststellen, dass die Verzögerung im Netz t_N nicht

```
Routenverfolgung zu www.tu-darmstadt.de [130.83.47.128]
über maximal 30 Abschnitte:
  1   <1 ms  corona.ibr.cs.tu-bs.de [134.169.34.1]
  2   <1 ms  134.169.39.254
  3   <1 ms  xwin-bs-gi4-4.rz.tu-bs.de [134.169.3.45]
  4    1 ms  188.1.46.145
  5    2 ms  xr-mag1-te1-1.x-win.dfn.de [188.1.144.245]
  6    4 ms  xr-pot1-te2-1.x-win.dfn.de [188.1.144.253]
  7    4 ms  zr-pot1-te0-0-0-0.x-win.dfn.de [188.1.145.162]
  8    9 ms  hbg-b2-link.telia.net [213.248.69.33]
  9    9 ms  hbg-bb1-link.telia.net [80.91.251.77]
 10   19 ms  ffm-bb1-pos7-0-0.telia.net [213.248.64.42]
 11   20 ms  ffm-b6-link.telia.net [80.91.254.161]
 12   19 ms  manda-01424-ffm-b2.c.telia.net [213.248.76.102]
 13   20 ms  ge-0-3-11.rt1.sm.tu.da.man-da.net [82.195.67.65]
 14   21 ms  ge-0-2-827.br-tud1.sm.tu.da.man-da.net [82.195.67.197]
 15   22 ms  www-tu.tu-darmstadt.de [130.83.47.128]
```

Abbildung 5.9: traceroute zwischen dem IBR und der TU Darmstadt

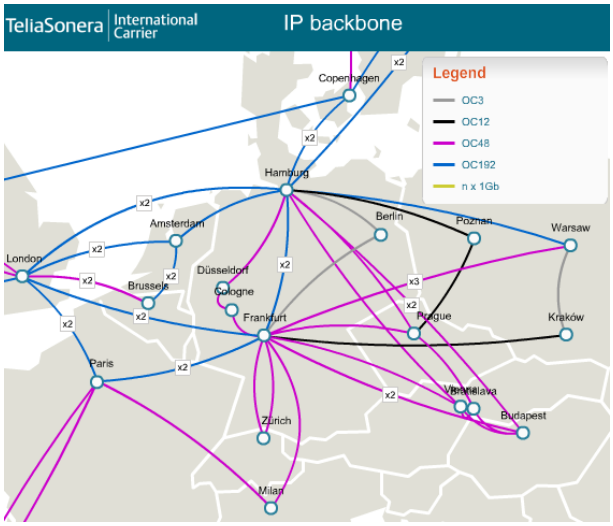

Abbildung 5.10: TeliaSonera Backbone

vorhergesagt werden kann – sie ist als statistische Verteilung $H(t)$ zu betrachten, die nach unten durch die Signalausbreitungszeit t_P über die Netzdistanz (oder hilfsweise der geografischen Distanz) zwischen den Endsystemen beschränkt ist. Die in den Routern zu erwartende Verzögerung variiert aufgrund der nicht deterministischen Wartezeit durch Queuing, die im Extremfall dann unendlich betragen kann, wenn ein Paket wegen Überlast im Router verworfen wird. Die Verteilung ist daher nach oben offen, eine schematische Darstellung von $H(t)$ ist in Abbildung 5.11 skizziert.

Die kumulative Verteilungsfunktion $p(t_d)$ mit

$$p(t_d) = P(H(t) \leq t_d) = \int_0^{t_d} H(t)\,dt \tag{5.11}$$

gibt Aufschluss darüber, mit welcher Wahrscheinlichkeit ein Paket innerhalb der Zeit t_d erfolgreich übertragen wird. Ist für eine Verbindung zwischen einem NMP-Client und Server diese Verteilungsfunktion bekannt, kann das Verhältnis von Wartezeit und Anteil erfolgreich übertragener Pakete bestimmt werden und dient dem Anwender als Werkzeug für die Parametrisierung des Systems. Er kann damit die Gesamtlatenz des Systems gegen die Paketverlustrate abwägen und so einen Kompromiss zwischen Verzögerung und Audioqualität finden.

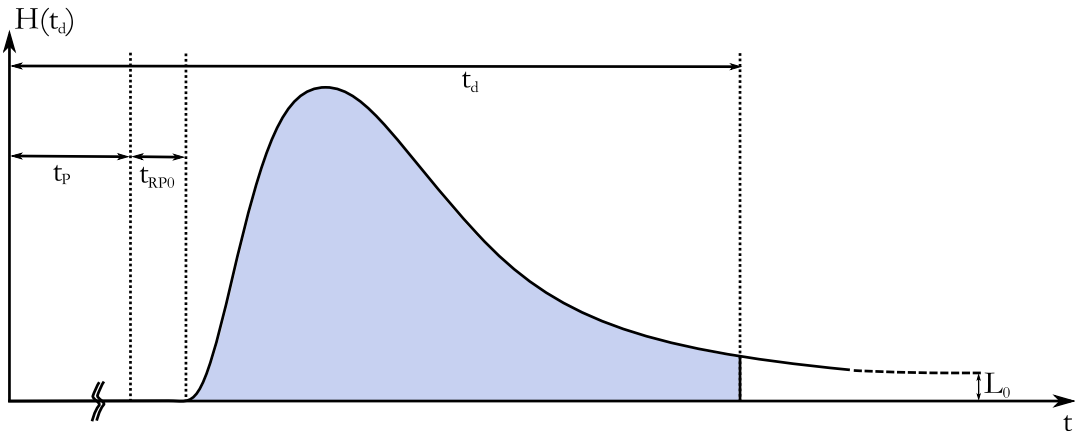

Abbildung 5.11: Verteilungsfunktion der Übertragungsverzögerung im Netz

5.8 Fazit

Die Ergebnisse dieser Analyse und der durchgeführten Messungen belegen, dass die Realisierung von NMP möglich sein sollte. Möchte man in einem solchen System die Gesamtlatenz unter 30 ms halten, ist der Anwendungsradius wegen der endlichen Signalausbreitungsgeschwindigkeit bereits aus physikalischen Gründen auf etwa 3000 km beschränkt. In der Praxis verweilen die Audiodaten bereits in den NMP Endsystemen (Client und Server) knapp 14 ms, Verzögerungen in den Netzknoten auf den Routen zwischen Client und Server senken den Einsatzradius weiter auf einen Wert von etwa 450 km Netzdistanz herab. Geografisch betrachtet sollten die Endsysteme nicht weiter als 300 km voneinander entfernt sein, um eine Gesamtlatenz unter 30 ms bei guter Audioqualität bieten zu können.

Bei der praktischen Umsetzung von NMP engt die erforderliche Einhaltung strikter Latenzgrenzen die Möglichkeiten stark ein und erfordert ein geradliniges Konzept, das diesen Einschränkungen Rechnung trägt. Ausgehend von der durchgeführten Analyse sind unterschiedliche Herausforderungen bei der Bewältigung dieser Aufgabe zu erfüllen:

Realisierung der Endsysteme mit minimaler Latenz

Wir haben hier idealisierte Annahmen verwendet, um die untere Schranke für die Verweildauer der Daten in den Endsystemen abzuschätzen. Diese Idealisierungen erschweren den Einsatz etablierter Verfahren und Methoden zur Realisierung ein oder verbieten diese sogar. Beispiele dafür sind effiziente Codierverfahren, komplexe Fehlerkorrekturen oder das erneute Senden im Netz verlorener Datenpakete. Bedingung für die Realisierbarkeit von NMP ist das Einhalten der maximalen Verzögerung in den Endsystemen von knapp 14 ms. In einer ersten Phase ist die praktische Umsetzbarkeit der Annahmen mit einem Prototypen nachzuweisen, der in einem nahezu idealen Netz diese Latenzgrenze nicht überschreitet.

Anpassung der Systemparameter an die Netzeigenschaften

Für die Durchführung einer NMP Sitzung müssen die Eigenschaften des Netzes hohen Anforderung hinsichtlich der mittleren Verzögerung und des tolerierbaren Jitters genügen. Für die Bestimmung der Eigenschaften vor dem Aufsetzen von Sitzungen und die kontinuierliche Anpassung der Betriebsparameter an eventuelle Schwankungen sind zuverlässige Methoden für die Bestimmung der Netzgüte und zur dynamischen Parametrisierung des Systems zu entwerfen. Das System muss Aussagen darüber treffen können, ob eine geplante Sitzung über das vorhandene Netz den Benutzeranforderungen gemäß zufriedenstellend durchführbar ist und während der Sitzung autonom und für Musiker transparent auf auftretende Änderungen der Netzcharakteristiken reagiert.

Realisierung von Fehlerrobustheit

Der inhärente Zusammenhang zwischen der Beschränkung der Netzverzögerung und der Paketverlustrate in IP-Netzen bedingt, dass es bei Vorgabe einer maximalen Latenztoleranz unweigerlich zu Paketverlusten in Form von zu spät eintreffenden oder verworfenen Paketen kommen muss. Die Endsysteme müssen also jederzeit mit Paketverlusten rechnen und gegenüber diesen robust ausgelegt sein. Der gesamte Datenpfad ist so umzusetzen, dass Paketverluste erkannt und bezüglich der einzuhaltenden Latenz und Audioqualität bearbeitet werden.

Im folgenden Kapitel widmen wir uns detailliert diesen Herausforderungen und beschreiben unsere Lösungsansätze.

KAPITEL 6

ZENTRALE SCHWIERIGKEITEN UND LÖSUNGSANSÄTZE

In den bisherigen Kapiteln haben wir die Besonderheiten eines NMP-Systems vorgestellt und einen Einblick in die spezifischen Schwierigkeiten gegeben. Die kritische Anforderung einer geringen Systemlatenz haben wir im vorhergehenden Kapitel im Detail betrachtet und basierend auf der durchgeführten Analyse und praktischen Messungen eine prinzipielle Machbarkeit von NMP ermittelt, die bei optimaler Realisierung gemeinsames Musizieren innerhalb eines definierten Radius ermöglicht.

In diesem Abschnitt stellen wir unsere Lösungsansätze für die vorgestellten Probleme vor und beschreiben alle für einen Systementwurf erforderlichen Maßnahmen. Der Fokus liegt dabei zunächst unverändert auf der Erfüllung der Kernanforderung geringer Latenz. Daran anschließend werden für alle in Kapitel 3 thematisierten Unwägbarkeiten Lösungen diskutiert und für die beschriebenen Abhängigkeiten untereinander Verfahren für die Bestimmung sinnvoller Kompromisse vorgestellt.

6.1 Generelle Vorgehensweise und Voraussetzungen

Aufgrund der in Kapitel 3 vorgestellten Herausforderungen gestaltet sich die Untersuchung und Realisierung von NMP als Gesamtsystem als höchst komplex.

Die Komposition aus Endsystemen und Netz führt zu einer Überlagerung unterschiedlicher Betriebsparameter und erschwert deren Kontrolle bei der Einhaltung vorgegebener Anforderungen. Zur Veranschaulichung betrachten wir beispielhaft die Latenz und den Jitter als zentrale Herausforderung auf dem Datenpfad zwischen Erzeugung und Verbrauch der Audiodaten im Client.

Es wurde gezeigt, dass bereits beim Zusammenspiel von Audiohardware und Betriebssystem eine minimale systematische Pufferverzögerung auftritt, die durch das nicht-deterministische Verhalten des Schedulers im Betriebssystem um einen variablen Anteil erhöht wird. Die daran anschließende Datenverarbeitung auf der Anwendungsschicht variiert ebenfalls, man denke dabei bspw. an die Audiocodierung, deren Komplexität von der Art der Audiodaten abhängt. Diese schwankende Verarbeitungszeiten treten beim Server multiplikativ auf, da mehrere Datenströme zu verarbeiten sind.

Die Latenzkomponente beim Datentransport im Netz nimmt eine Sonderrolle ein: sie trägt den bei Weitem höchsten Anteil der Variabilität hinzu und ist darüber hinaus nicht beeinflussbar. Damit stellt das Netz eine systemimmanente und unüberwindbare Trennung zwischen den Endsystemen dar.

Bei der Realisierung von NMP haben wir zwei generelle Ansatzweisen umgesetzt und untersucht. Da die Lösungsmöglichkeiten für die beschriebenen Herausforderungen vom gewählten Ansatz abhängig sind, werden diese beiden Varianten an dieser Stelle kurz vorgestellt.

6.1.1 Gesamtsystem mit durchgehender Jitter-Kontrolle

Grundlage jeder Echtzeit-Streaminganwendung allgemein und bei NMP im besonderen ist die Regel, dass Daten, die zum Abspielzeitpunkt noch nicht verfügbar sind, bei einem späteren Eintreffen nutzlos sind und verworfen werden. Verspätet kommen Daten dann an, wenn die Übertragungszeit des betreffenden Paketes die Abspielverzögerung übersteigt. Die Dimensionierung der Abspielverzögerung stellt eine Balance zwischen Latenz und Paketverlustrate dar, die so gewählt wird, dass sie die mittlere Übertragungsverzögerung um eine De-Jitter Reserve erweitert. Die maximal tolerierbare Latenz legt bei vorgegebenen Netzeigenschaften somit implizit den Anteil verspätet eintreffender Pakete fest.

Bei NMP ist diese Balance zentraler Bestandteil, denn die geringe Latenztoleranz erfordert, dass einerseits alle beeinflussbaren Einzelkomponenten auf minimale Latenz getrimmt werden müssen und andererseits eine relativ hohe Paketverlustrate vorherrscht. Wegen der geringen Verfügbarkeit zusätzlicher De-Jitter Reserven gewinnen die auf den übrigen Datenpfad auftretenden Latenzschwankungen in Relation zum Netz-Jitter an Bedeutung, die üblicherweise vernachlässigt werden.

Die Grundidee bei der Umsetzung als Gesamtsystem mit durchgehender Jitter-Kontrolle ist daher folgende: wenn durch die Beschränkung der Länge des Netz-De-Jitter Puffers die Schwankungen der Latenz bei der Datenverarbeitung in ähnliche Größenordnungen steigt, müssen für eine effektive Behandlung alle Komponenten auf dem Datenpfad kontrolliert werden. Denn dann ist die Verzögerung eines Datenpaketes bis nach seinem Abspielzeitpunkt nicht mehr wesentlich durch den Netz-Jitter verursacht worden. Vielmehr ist nach jeder Jitter verursachenden Funktionseinheit zu prüfen, ob das gerade bearbeitete Paket nicht bereits so stark verzögert wurde, dass es proaktiv verworfen werden kann, da es zum Abspielzeitpunkt nicht rechtzeitig ausgeliefert werden kann.

Zur besseren Veranschaulichung blicken wir erneut auf den Datenpfad und greifen uns beispielhaft einige Stationen heraus, an denen eine Prüfung sinnvoll ist:

1. Verzögerungen beim Aufruf der ISR können dazu führen, dass Datenblöcke verspätet von der Audiohardware ausgelesen werden. Übersteigt die Verspätung die verfügbaren De-Jitter Puffer, können die betreffenden Daten bereits hier verworfen werden.
2. Kann der Scheduler die nachrangige Bearbeitung auf der Anwendungsschicht wegen anderer gleichzeitig aktiver Tasks nicht sofort starten, treten beim Taskwechsel Verzögerungen auf, die u.U. über der verfügbaren Reserve liegen, womit das betreffende Paket zum Abspielzeitpunkt voraussehbar verspätet vorliegen wird und daher unbrauchbar wird.
3. Ebenso kann mit einer Messung der Übertragungsdauer vom Client zum Server ermittelt werden, ob sich die bisher aufgefallenen Latenzen soweit aufsummiert haben, dass eine weitere Bearbeitung des Paketes sinnlos wird.

Auf dem rückläufigem Pfad ergeben sich analoge Prüfstationen, an denen anhand der bisherigen kumulierten Latenz eine Entscheidung über die weitere Verwertung des jeweiligen Paketes getroffen werden kann. Der Gewinn eines solchen proaktiven Ansatzes ist naheliegend: je früher verspätete Pakete identifiziert und verworfen werden können, desto ressourcenschonender ist die Annullierung der nicht verwertbaren Daten. Ein bereits beim Auslesen aus der Audiohardware verworfener Datenblock spart Verarbeitungszeit beim Client und Server und Netzlast für die Übermittlung zum und vom Server.

Diesen quasi-optimalen Ansatz haben wir beim NMP anfangs umgesetzt, da er im Labor praktikabel ist. Im lokalen Netz können die Rechneruhren der verwendeten Endsysteme genau genug synchronisiert werden (mehr dazu in Abschnitt 6.6), so dass durchgängig eine gemeinsame Zeitdomäne mit durchweg gültigen Zeitstempeln verwendet wird. Aus den Zeitstempeln für Versende- und Empfangszeitpunkt eines Paketes kann so die genaue Übertragungszeit im Netz ermittelt werden.

In Weitverkehrsnetzen ist dagegen die erreichbare Zeitsynchronisation in der Größenordnung von 10m̀s nicht ausreichend genau, um aus Zeitstempeln über Zeitdomänen hinweg eindeutig unterscheidbare Latenzkomponenten berechnen zu können. So lassen sich zwar innerhalb eines Endsystems die Verzögerungen einzelnen Stationen zuordnen, wenn aber die Übertragungslatenz als Hauptkomponente bei der Betrachtung fehlt, ist ein vorauseilender Datenverwurf nicht mehr praktikabel.

Neben dieser beschränkten Verwendbarkeit, der höheren Komplexität und des damit einhergehenden höheren Ressourcenbedarfs haben wir diesen Ansatz aufgrund funktionaler Anforderungen ersetzt. Die in dieser Arbeit nicht im Detail diskutierte *Rehearsal On-Demand* Funktionalität, also die Fähigkeit unseres NMP-Systems, aufgezeichnete Sitzungen abzurufen oder virtuelle Clients zu simulieren, erfordert, dass alle von den Clients produzierten Daten beim Server ankommen. Pakete dürfen daher beim Client nicht verworfen werden, auch wenn dort bereits erkennbar ist, dass sie in der laufenden Sitzung nicht ausgespielt werden können. Unter dieser Vorgabe verbleibt nur noch der Verwurf von Daten vor der Rücksendung beim Server als sinnvoller proaktiver Eingriff.

Das dabei erzielbare Einsparpotenzial hat sich allerdings als so gering erwiesen, dass der Mehraufwand für die Verwaltung und den Transport der Zeitstempel in keiner Weise gerechtfertigt wird.

6.1.2 Einzelkomponenten mit zentraler Jitter-Kontrolle

Ungeachtet der potentiellen Möglichkeit, durch vorauseilenden Datenverwurf Ressourcen zu schonen, gibt es tatsächlich nur eine Komponente im NMP-System, die eindeutig darüber befinden kann, ob ein Audiopaket zum Abspielzeitpunkt rechtzeitig vorhanden ist: die Audiohardware. Erst wenn sie über den ISR ein Audiopaket zum Ausspielen anfordert und dieses bis zu eben diesen Zeitpunkt nicht vorhanden ist, erst dann herrscht Gewissheit darüber, dass dieses Paket verworfen werden kann, wenn es zu einem späteren Zeitpunkt empfangen werden sollte.

Im Umkehrschluss bedeutet das, dass das oben beschriebene und nicht mehr verwendete proaktive Verfahren bei jeder vorauseilenden Aktion spekulativ handelt – es kann möglicherweise Pakete verwerfen, die noch rechtzeitig angekommen wären, weil die Latenz auf dem restlichen Weg besonders gering ausfiele.

Aus dieser Überlegung heraus haben wir beim aktuell verwendeten Verfahren die Audiohardware als zentrale Komponente für die Jitter-Kontrolle gewählt und verzichten auf jegliche zusätzliche Prüfung auf dem gesamten Datenpfad. Damit wird eine drastische Vereinfachung des Systemdesigns realisierbar, die viele der in diesem Abschnitt diskutierten Lösungsansätze ermöglicht.

Die Vereinfachung rührt aus einer damit ermöglichten Auftrennung des Gesamtsystems in einzelne Komponenten, die wir bei der Untersuchung als abgeschlossene Teilsysteme betrachten und damit die Komplexität aufbrechen können. Für die Komponenten definieren wir Anforderungen an Funktion und Verhalten derart, dass diese einfach umzusetzen und nachzuweisen sind und im Effekt das zusammengesetzte System ähnlich optimale Eigenschaften bietet, wie das bisher betrachtete Gesamtsystem.

Die Aufteilung ergibt sich intuitiv mit dem Netz als nicht beeinflussbare Funktionseinheit, die Client und Server als weitere Systemkomponenten voneinander trennt. Client und Server werden in diesem Design als abgeschlossene *Black Boxes* betrachtet, die definierten Anforderungen genügen müssen und miteinander einzig über die ausgetauschten Pakete als Datenschnittstelle interagieren.

Folgende Anforderungen werden dabei für den Client aufgestellt:

1. Audiodaten müssen so schnell wie möglich von der Audiohardware angenommen, verarbeiten und am Ausgang dem Netz übergeben werden
2. die am Eingang vom Netz empfangenen Daten müssen verarbeitet und mit einer geeigneten Abspielverzögerung der Audiohardware zugeführt werden

Am Ausgang eines idealen Clients werden Audiopakete in äquidistanten Abständen von t_φ ausgegeben und im Netz versendet. Ein gleiches, jedoch um eine konstante Übertragungs- und Verarbeitungslatenz versetztes, Paketmuster wird idealerweise am Eingang vom Netz empfangen. In realer Umgebung kommen die Antwortpakete am Eingang mit variabler Paketzwischenankunftszeiten an und müssen mit einer geeigneten Abspielverzögerung ausgegeben werden. Diese kann dann einfach berechnet werden, wenn das Alter eines Antwortpaketes vom Client jederzeit eindeutig bestimmbar ist.

Auch der Server kommt mit lediglich zwei Vorgaben für die Verarbeitung von Paketen aus, um das optimale Verhalten des zuvor betrachteten Gesamtsystems zu erreichen: zeitlich zusammengehörige Audiopakete aller Clients werden verarbeitet, gemischt und zurückgesendet und zwar

1. so früh wie möglich, d.h., unmittelbar wenn von jedem Client das korrespondierende Paket vorliegt
2. zur Kompensation von Jitter im Client und Netz wird eine geeignete Mischverzögerung verwendet, die die maximale Aufenthaltsdauer eines Paketes im Server nach oben limitiert.

Analog zum Client wird der Server als Black Box angesehen, der mit jedem Client einer Sitzung je eine eingehende und eine ausgehende Datenverbindung unterhält. Im Idealfall laufen auf jedem Eingang die Pakete isochron ein und werden nach spätestens t_φ als gemischtes Paket wieder zurückgesendet. Real variieren die Paketzwischenankunftszeiten, die zu verwendende Mischverzögerung dient als De-Jitter Puffer.

Ein derart modulares Design mit einer lokal beschränkten Untersuchung einzelner Aspekte und fokussierter Problemlösung macht die Umsetzung von NMP erst praktisch beherrschbar. Für die Realisierung des kombinierten Systems ist, neben den Anforderungen für die Einzelkomponenten, nur eine Bedingung einzuhalten: jeder Client muss jedes Antwortpaket des Servers eindeutig dem zuvor selbst generierten und versendeten Paket zuordnen können. Nur damit sind modulare Funktionseinheiten umsetzbar, wie bspw. Dimensionierung von Abspiel- und Mischpuffers.

Praktisch wird diese Bedingung dadurch erreicht, dass

- jedes Audiopaket bei der Generierung in der Soundkarte vom Client mit einer Sequenznummer versehen wird
- diese Sequenznummer über den gesamten Lebenszyklus des Paketes bis zur Ausgabe über die Audiokarte gleich bleibt

In den folgenden Abschnitten werden wir die wichtigsten Herausforderungen unseres NMP-Systems untersuchen und dabei implizit dieses modulare Systemdesign als Basis annehmen. Die persistente Sequenzierung von Audiopaketen wird als gegeben angenommen.

6.2 Mischen

Das Kombinieren der einzelnen Audiodatenströme zu einem Mischsignal ist die zentrale Aufgabe des Servers und die Systemkomponente, mit der die Telepräsenz generiert wird. Das Bereitstellen einer gemeinsamen Audiospur hebt unser NMP von allen vergleichbaren Verfahren ab, die nur eine Ende-zu-Ende Übertragung vornehmen. Die Grundlagen des Mischens digitalen Audios mit den inhärenten Probleme bei der Überlagerung einer großen Anzahl von Audiospuren werden in diesem Abschnitt diskutiert. Darüber hinaus definieren

wir hier, welche Formen des Mischens wir für die Erzeugung unterschiedlicher Immersionsgrade unterstützen wollen und beschreiben die daraus resultierenden Anforderungen für unser NMP-System.

6.2.1 Mischen von Audio in realen Musikumgebungen

Als Grundlage für die Simulation virtueller Bühnen in NMP verwenden wir das Mischen von Audiodaten in realen Umgebungen, wie es bei der Aufzeichnung oder Beschallung auf Bühnen und in Musikstudios praktiziert wird. Wir schauen uns daher zunächst an, wie die Realisierbarkeit eines akustischen Mischens aussieht, und gehen dann über zum digitalen Mischen.

Für unser Ansinnen, eine virtuelle Bühne zu simulieren, müssen wir uns als erstes fragen, wie genau wir die Wirklichkeit abbilden können. Richten wir unsere Augen auf drei Musiker während einer Probe auf der Bühne. Der Höreindruck bei jedem von ihnen ist unterschiedlich und von unzähligen Parametern abhängig. Dabei spielt der Raum wegen der Schallreflexionen eine Rolle, genauso wie die Streuungen der Schallwellen an Gegenständen und Personen im Raum. Mit welcher Intensität und Verzögerung sich die Musiker untereinander hören, ist im Wesentlichen abhängig von ihrer Position zueinander.

In unserer Anwendung bewegen wir uns daher abseits dieser Ansätze hoher Immersion und beschränken uns auf die praktikablen und in der Musikproduktion eingesetzten Verfahren, wie sie bei der Produktion und Aufnahme von Live-Musik auf Bühnen und in Studios verwendet werden. In solchen Aufbauten wird das Audiosignal jedes teilnehmenden Musikers abgegriffen, zentral gemischt und das Mischsignal dann den Musikern (und evtl. dem Publikum) zurückgesendet.

So einfach dieses Grundprinzip sich auch anhört – die erforderlichen Vorkehrungen für einen guten Klang sind enorm. Das Abmischen der unterschiedlichen Audiospuren, auch *Soundcheck* genannt, geht jeder musikalischen Veranstaltung voran. Dabei werden die Signalpegel der Musiker in Abhängigkeit von den akustischen Gegebenheiten am Veranstaltungsort und dem angestrebten auditiven Eindruck beim Publikum abgemischt. Das Vorgehen dabei ist fest vorgegeben: zunächst prüft jeder Musiker sein direktes Feedback, indem er seine lokale Verstärkung individuell so einstellt, dass er sein Instrument wie gewohnt wahrnimmt. Im nächsten Schritt wird das Mischsignal von einem Tonmeister abgestimmt. Dabei werden die Anteile der Tonspuren zueinander und die Klangspektren abgeglichen und akustische Besonderheiten der Umgebung berücksichtigt.

Nach dem Soundcheck nimmt der Musiker beim Spielen jeweils zwei Audiosignale wahr. Sein direktes Feedback, das im Falle von akustischen Instrumenten direkt gehört und bei elektrischen Geräten unverfälscht vom Verstärker wiedergegeben wird, erlaubt das gewohnte Spielen des jeweiligen Instrumentes. Das Mischsignal vermittelt den Akteuren hingegen den für die Zuhörer vorbereiteten Musikeindruck, der aufgrund der Parametrisierung des Mischers stark vom direkten Feedback abweichen kann.

Diese Komposition des Feedback- und Mischsignals beim Musiker müssen wir in unserem System unterstützen, um einen vergleichbaren und für den Anwender akzeptablen Höreindruck vermitteln zu können. Als Forderung geht daraus für den NMP-Client hervor, dass er jedem Anwender die Möglichkeit bieten muss, das Feedbacksignal zu parametrisieren und mit dem vom NMP-Server erhaltenen Mischsignal zu überlagern. Dabei müssen sowohl die Anteile wie auch der zeitliche Versatz der beiden Signale zueinander angepasst werden können. Der NMP-Server muss dagegen die Parametrisierung des Mischers für die Erzeugung des Mischsignals unterstützen.

6.2.2 Digitales Mischen von Audiodaten

Die Superposition von Schallwellen wird beim digitalen Mischen von Audiodatenströmen durch die arithmetische Addition der zeitsynchronen Abtastwerte abgebildet. Werden zwei Signale mit identischer zeitlicher

und räumlicher Auflösung abgetastet und liegen in den Samples a_i und b_i vor, berechnen sich die Abtastwerte des Mischsignals als Summe der Amplituden $m_i = a_i + b_i$. Die aus der Akustik bekannten Eigenschaften der Interferenz behalten ihre Gültigkeit auch beim digitalen Mischen, es treten konstruktive und destruktive Interferenzen auf die zur Verstärkung bzw. Auslöschung von überlagerten Signalen führen können.

Die Verstärkung ist beim Mischen von Audio allgemein und im digitalen Fall besonders problematisch, da sie zu Überläufen führen kann. Die Summe der Amplituden zweier Samples, die jeweils einen durch die räumliche Auflösung definierten maximalen Betrag nicht überschreiten, kann ohne Weiteres diesen Wertebereich übersteigen. Die Amplitude muss dann für dieses Sample im Mischsignal auf den Maximalwert limitiert werden, man spricht von *Clipping*. Die resultierende Diskontinuität des Signalverlaufs macht sich bei der Audioausgabe als Verzerrung sehr störend bemerkbar.

Eine dauerhafte Übersteuerung muss daher durch geeignete Maßnahmen vermieden werden. Dafür stehen die Ansätze der statischen und dynamischen Verstärkungsanpassung zur Verfügung, um die Amplituden des Mischsignals im gültigen Wertebereich zu halten.

6.2.2.1 Proaktive Vermeidung von Clipping durch statische Signalabschwächung

Bei der Überlagerung von Audiosignalen wird der maximale Pegel erreicht, wenn die zu mischenden Samples jeweils den maximalen Wert haben, wenn also gilt $a_c = b_c = l_{max}$ und die Amplituden sich zum doppelten Maximalwert addieren, formal $m_c = a_c + b_c = 2 \cdot l_{max}$. Clipping kann offensichtlich sicher verhindert werden, wenn die Samples des gemischten Signals in ihrer Amplitude halbiert werden, so dass auch im äußersten Fall der gültige Wertebereich nicht verlassen wird. Analog gilt, dass beim Mischen von n_C Audiosignalen eine Abschwächung des Mischsignals um Faktor n_C^{-1} erfolgen muss, um Clipping sicher zu vermeiden. Formal gilt dabei

$$m_n = \frac{\sum_{i=1}^{i=n_C} a_{n_i}}{n_C}$$

So zuverlässig diese statische Abschwächung ist, so störend kann sie bei der Verwendung in NMP in zweierlei Hinsicht sein. Das naheliegende Problem tritt beim dynamischen Aufbau einer Sitzung und der einleitenden Pegelkonfiguration der Teilnehmer untereinander auf. Jede Änderung der Zusammenstellung der Sitzung macht sich bei einer statischen Anpassung der Signalabschwächung des Mischsignals deutlich bemerkbar. Das Signal des ersten Spielers würde sich unmittelbar auf die Hälfte abschwächen, sobald sich der zweite Spieler in die Sitzung einklinkt. Ein bis dahin durchgeführter Soundcheck müsste bei jeder Änderung der Zusammenstellung wiederholt bzw. angepasst werden. Diese Unzulänglichkeit lässt sich durch Einschränkungen der Freiheiten beim Soundcheck und dem Sessionaufbau bspw. derart beheben, dass die Pegelparametrisierung erst nach dem vollständigen Beitritt aller Teilnehmer zur Sitzung erfolgt und eine folgende Änderung der Zusammenstellung unterbunden wird.

Das schwerwiegendere Problem tritt bei höheren Teilnehmerzahlen n_C auf, da durch die Pegelabschwächung der Dynamikbereich signifikant eingeschränkt wird. Nimmt man zur Verdeutlichung eine Session mit acht Musikern an, beträgt der Pegelanteil jedes Teilnehmers im Mischdatensignal nur ein Achtel seiner ursprünglichen Aussteuerung. Von den 16 Bits für die räumliche Auflösung gehen nach der Skalierung des Mischsignals drei Bits verloren, so dass effektiv mit 13 Bits gearbeitet wird. Neben einer generellen Absenkung des Signal-Rausch Abstandes führt die Skalierung dazu, dass leisere Instrumente dann evtl. überproportional verstärkt werden müssen, um im Mix ausreichend wahrgenommen zu werden. Die durch die Maßnahme verursachte Absenkung der mittleren Lautstärke kann am Client mit einer Erhöhung des Ausgabepegels kompensiert werden, was aber letztlich die Gefahr impliziert, den Wertebereich des DAU in der Audiohardware zu übersteuern.

Diese Schwierigkeiten sind der garantierten Vermeidung des Clippings geschuldet. Bei einem Verzicht auf eine absolute Sicherheit eröffnet sich Spielraum, die Auftrittswahrscheinlichkeit von Verzerrungen gegen die Höhe der Signalabschwächung abzuwägen. Eingangs haben wir erwähnt, dass ein dauerhaftes Übersteuern als sehr störend empfunden wird und anschließend erörtert, wann Clipping überhaupt entsteht. Zwei Aspekte verdeutlichen dabei, dass eine absolute Sicherheit überdimensioniert sein kann. Zum einen sind die Eingangspegel der einzelnen Teilnehmer üblicherweise so eingestellt, dass eine Aussteuerung von 25 bis 50% (entsprechend -6 bis -3 dB) erreicht und eine Reserve vorgehalten wird, um auch die lautesten Passagen verzerrungsfrei auflösen zu können. Diese bei jedem Teilnehmer vorhandenen Reserven können bei der Skalierung beim Mischen berücksichtigt werden.

Der zweite zu berücksichtigende Faktor ist die Eintrittswahrscheinlichkeit für Clipping. Zwar bestimmen Takt und Rhythmus die Musik und geben vor, dass alle Teilnehmer in unmittelbarer zeitlicher Nähe Töne generieren. Dass jedoch alle gleichzeitig und Sample genau die maximale Amplitude erreichen, ist höchst unwahrscheinlich.

Ein bewährter Ansatz, beide Aspekte bei der Skalierung des Mischsignals zu berücksichtigen, ist die Verwendung der geometrischen statt der arithmetischen Anpassung. Das Signal wird also nicht mehr um Faktor n_C^{-1} abgeschwächt, sondern um $\sqrt{n_C}$. Formal gilt dann für die Berechnung der Samplewerte

$$m_n = \frac{\sum_{i=1}^{i=n_C} a_{n_i}}{\sqrt{n_C}}$$

Damit wird der Dynamikbereich des Mischsignals deutlich weniger abgeschwächt und erhöht sich mit steigender Teilnehmerzahl n_C nur moderat, verhindert dafür im Gegenzug jedoch das Auftreten von Clipping nicht sicher. Bei der Bestimmung des Skalierungsfaktors sind darüber hinaus beliebige Freiheiten möglich, um eine Balance zwischen absolutem und teilnehmerabhängigen Anteil zu erreichen. Es lassen sich so auch komplexe Abhängigkeiten des Faktors von der Teilnehmerzahl herstellen (bspw. durch eine polynomiale), dabei findet immer eine Abwägung zwischen Auftrittsrisiko einer Übersteuerung und Einschränkung des Dynamikbereichs im Mischsignal statt.

6.2.2.2 Reaktive Vermeidung von Clipping durch automatische Verstärkungsregelung

Das Problem einer unausgewogenen Aussteuerung ist nicht auf das Mischen beschränkt, es ist vielmehr ein alltägliches Phänomen. Wer hat sich nicht schon darüber geärgert, dass beim Fernsehen die Lautstärke in der Werbepause deutlich höher ist als im Film und manuell verstellt werden muss? Gleiches gilt für die schlechte Verständlichkeit des Telefonpartners, wenn sich sein Abstand zum Mikrofon ändert. Als Lösungsansatz für dieses Problem hat sich die automatische Verstärkungsregelung AGC (*Automatic Gain Control*) etabliert.

AGC beruht auf einer kontinuierlichen Messung des Audiosignals und einer reaktiven Anpassung des Pegels, um eine voreingestellte mittlere Lautstärke durchgängig einzuhalten. Dabei werden die Audiodaten in Blöcke konstanter Länge unterteilt und die Lautstärke für jeden Block so angepasst, dass ein vorgegebener Bereich nicht verlassen wird. Die Länge der verwendeten Blöcke legt dabei die Reaktionsgeschwindigkeit und die Wahl der Toleranzen die Empfindlichkeit des Verfahrens fest. Für die sichere Unterscheidung, ob eine lautere Passage beabsichtigt ist und somit nicht korrigiert werden darf, oder durch eine fehlerhafte und zu korrigierende Aussteuerung erfolgt ist, müssen längere Blöcke analysiert werden - und das bevor sie ausgegeben werden. Das führt dazu, dass AGC nur in zeitunkritischen Anwendungen sinnvoll eingesetzt werden kann, in denen genügend Zeit für die sichere Erkennung von korrigierbaren Lautstärkeschwankungen verfügbar ist. In interaktiven Anwendungen wie der Telefonie wirkt sich eine empfindliche AGC eher störend aus und wird deshalb oft optional eingesetzt.

In der Musik ist AGC komplett verpönt, aus ganz offensichtlichem Gründen. Die Dynamik ist hier immer beabsichtigt und Teil der Musikparameter, eine Verstärkung leiser Passagen würde bspw. den Kontext des Musikstücks verändern. Verwendet wird auch in der Musik jedoch das Normalisieren, was formal als eine extreme Form des AGC interpretiert werden kann. Die Blockgröße wird dabei auf die gesamte Länge des Musikstücks ausgedehnt und eine optimale Aussteuerung bestimmt. Dafür wird eine konstante Skalierung des Signals entweder derart gewählt, dass die größte auftretende Signalamplitude mit dem höchsten Wert aus dem vorhandenen Wertebereich zusammenfällt, oder die mittlere Lautstärke einen vorgegebenen Wert annimmt. In der Produktion von Musik ist die Normalisierung ein integraler Anteil des Masterings, bspw. bei der Herstellung von Audio-CDs.

Die Normalisierung ist nur in der Nachbearbeitung möglich, sobald alle Audiodaten vorliegen und eine konstante Skalierung möglich ist. Eine dynamische Anpassung während des Musizierens, die einen Soundcheck ersetzen könnte, ist weder möglich noch erwünscht.

Als Konsequenz für den Einsatz von AGC in NMP ist zu berücksichtigen, dass ein autonom agierendes System zur dynamischen Anpassung der Aussteuerung keine Akzeptanz bei Musikern finden wird. Als begleitende Maßnahme zur Vermeidung von Verzerrungen kann es in Erwägung gezogen werden, muss dann aber in jedem Fall abschaltbar sein.

6.2.2.3 Kombinierter adaptiver Ansatz

Wir sehen, dass beide beschriebenen Verfahren je einen Extremwert der Pegelanpassung darstellen – das proaktive verhindert Übersteuerungen sicher, jedoch auf Kosten einer verringerten Dynamik, das reaktive verwendet eine optimale Dynamik auf Kosten kontinuierlicher und für Musiker störender Lautstärkeänderungen.

Für unser NMP haben wir einen adaptiven Ansatz gewählt, der die beiden Verfahren so kombiniert, dass eine Session spezifische Dynamik sicher gestellt und dabei auf eine kontinuierliche Anpassung verzichtet wird. Grundlage dieses Ansatzes ist ein Ablauf in drei Phasen:

1. Wahl des Skalierungsfaktors unter optimistischen Annahmen
2. Pegelverfolgung und Detektion von Übersteuerungen
3. Anpassung des Skalierungsfaktors nur in eine Richtung (Abschwächung)

Beim Aufstarten einer Session wird im Mixer zunächst unter einer optimistischen Annahme ein Skalierungsfaktor von s_{n_C} angenommen. Da Fließkommaarithmetik auch bei heutigen Prozessoren nicht mit der Geschwindigkeit bei Ganzzahlen mithalten kann, wird für eine effiziente Berechnung dieser Wert gerundet und der resultierende Faktor si_{n_C} verwendet. In Prozessorarchitekturen, bei denen darüber hinaus arithmetische Operationen deutlich langsamer ablaufen als Bitoperationen, werden diese stattdessen auf die nächste Zweierpotenz sb_{n_C} gerundet, um eine effiziente Berechnung durch reines Shiften zu erreichen. Es ergibt sich damit folgende Tabelle für die initiale Skalierung der Samples:

Ausgehend von dieser initialen Skalierung beobachtet der Mixer die Aussteuerung des gemischten Signals kontinuierlich und detektiert Ereignisse, die zu Übersteuerungen führen. Die Entscheidung, unter welchen Umständen solche Ereignisse zu einer Anpassung des Skalierungsfaktors führen, ist anhand verfügbarer Messwerte und Statistiken parametrisierbar. Im einfachsten Fall kann jede detektierte Übersteuerung eine Anpassung nach sich ziehen, was jedoch offensichtlich nicht immer sinnvoll ist. So kann bspw. das Einklinken eines Instrumentes im elektrischen Abnehmer einen Pegelvollausschlag beim Client auslösen, sollte jedoch nicht zu einer Absenkung der Skalierung beim Mixer führen.

Sinnvoller ist es, anhand der Statistiken für mittlere Lautstärke, Anzahl der Übersteuerungen und deren zeitlichen Verlauf eine Vorhersage über künftige Ereignisse zu treffen und bei Bedarf eine Herabsenkung des

Teilnehmerzahl n_C	$s_{n_C} = \sqrt{(n_C)}$	si_{n_C}	sb_{n_C}
2	1.41	1	1
3	1.73	2	2
4	2	2	2
5	2.24	2	2
6	2.45	2	2
7	2.65	3	2
8	2.83	3	2
9	3	3	2
10	3.16	3	4

Skalierungsfaktors zu initiieren. Die aktuellen Standardeinstellungen im NMP verwenden hierbei bspw. die Bedingungen von fünf Ereignissen innerhalb von zwei Sekunden oder zehn Ereignissen innerhalb von zehn Sekunden.

Führt eine solche Entscheidung anschließend zu einer Anpassung der Skalierung, betritt der Mixer den Adaptionsmodus. Anhand der zurückliegenden Clippings wird zunächst ein neuer Zielwert für die Skalierung bestimmt, der mit einer vorgegebenen Wahrscheinlichkeit das Auftreten weiterer Übersteuerungen verhindert.

Ein unmittelbares Umschalten der Skalierung vom bisherigen in den neu berechneten Wert macht sich als Diskontinuität deutlich und störend bemerkbar. Daher wird ein Kontinuierlicher Übergang zwischen den beiden Werten durchlaufen. Über einen Zeitraum parametrisierbarer Länge ändert der Mixer seine Arbeitsweise, indem die Beschränkung auf ganzzahlige Operation aufgehoben wird und die Skalierung allmählich zwischen altem und neuem Wert wandert.

Abhängig von der vorgegebenen Adaptionszeitspanne t_a und der verwendeten Samplingrate f_s sind $n_{as} = t_a \cdot f_s$ Abtastwerte von der Anpassung betroffen. Als Defaultwert wird im NMP momentan eine Anpassungsdauer von 10 Sekunden verwendet, damit betrifft der Vorgang bei einer Abtastrate von 48 kHz $n_{as} = 480000$ Samples. Über diese durchläuft der Skalierungsfaktor die Zwischenschritte vom alten Wert s_{old} bis zum neuen Wert s_{new}.

Der Übergang erfolgt mit einer zu definierenden Adaptionsfunktion $af(i)$, die einen Wertebereich von $[0; 1]$ hat und für jedes Sample im Adaptionszeitraum den normierten Abstand zwischen s_{old} und s_{new} zurückgibt. Der Skalierungsfaktor s_i für jedes Sample im Adaptionszeitraum $i \leq n_{as}$ berechnet sich damit als

$$s_i = af(i) \cdot s_{new} + \big(1 - af(i)\big) \cdot s_{old}$$

Die Funktion ist so zu wählen, dass ein kontinuierlicher Übergang zwischen Null und Eins erfolgt. Sinnvollerweise sollte sie streng monoton sein und so skaliert werden, dass der Übergang über den gesamten Definitionsbereich von n_{as} Samples verläuft. Prinzipiell lässt sich der Verlauf von $af(i)$ frei parametrisieren, sinnvoll und tatsächlich unterscheidbar sind wenige Übergänge. Von diesen sind in NMP Verläufe für den linearen und einen sigmoiden (S-förmigen) Übergang enthalten.

Der lineare Verlauf verbindet Minimal- und Maximalwert über den Definitionsbereich, die resultierende Adaptionsfunktion lautet $af_l(i) = \frac{i}{n_{as}}$. Die Anpassung der Skalierung erfolgt beim linearen Übergang gleichbleibend über den gesamten eingestellten Zeitraum. Liegt dieser wie hier beschrieben in einer Größenordnung von 10 Sekunden, wird die Adaption auditiv wahrgenommen, und das über den gesamten Zeitraum.

Abhängig von der Art der Musik oder von subjektiven Präferenzen kann die wahrgenommene Beeinträchtigung durch die Wahl eines sigmoiden Übergangs gemindert werden. Dieser Verlauf zielt darauf ab, die Adaption zu Beginn und am Ende des Vorganges abzuschwächen. Um innerhalb des vorgegebenen Zeitraums die erforderliche Anpassung in selbem Umfang zu erreichen, muss diese Abflachung im Gegenzug

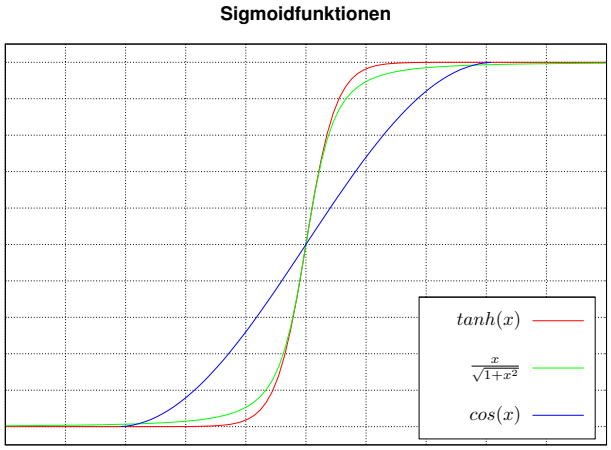

Abbildung 6.1: Sigmoide Adaptionsfunktion

durch eine größere Steigungen in der Mitte der Adaption kompensiert werden. Als sigmoide Funktionen kommen verschiedene bekannte Vertreter in Frage, wie bspw. die in Abbildung 6.1 dargestellten

- Tangens Hyperbolicus: $tanh(x) = \frac{e^x-1}{e^x+1}$
- einfache Funktionen wie: $\frac{x}{\sqrt{1+x^2}}$
- Teilabschnitte von Kreisfunktionen wie: $cos(x)$ mit $-\pi \leq x \leq 0$

Für unser NMP verwenden wir wegen der einfachen Berechenbarkeit den zuletzt genannte Teilabschnitt der Cosinusfunktion als Basis, die gemäß Definitions- und Wertebereich verschoben und skaliert wird.

Die Anpassungen auf der y-Achse sind intuitiv und unmittelbar ausführbar, da sie aus einer Translation der Funktion in y-Richtung um +1 und einer anschließenden Skalierung um den Faktor 0.5 besteht. Die Anpassung in x-Richtung ist von n_{as} und damit der Adaptionszeit und von der verwendeten räumlichen Auflösung der Abtastwerte abhängig. Für die Skalierung wird der gültige Wertebereich π auf die Anzahl der Samples im Adaptionsfenster n_{as} aufgeteilt und verschoben. Damit ergibt sich für unsere auf die Kreisfunktion beruhende sigmoide Adaptionsfunktion

$$af_s(i) = \frac{cos\left(\left(\frac{i}{n_{as}} - 1\right) \cdot \pi\right) + 1}{2} \qquad \text{mit} \qquad 0 \leq i \leq n_{as}$$

6.2.3 In NMP unterstützte Mischverfahren

In diesem Abschnitt wollen wir unterschiedliche Arten des Mischens von Audiodaten diskutieren und die in NMP implementierten Verfahren genauer erläutern.

6.2.3.1 Lineares Mischen

Unter dem linearen Mischen verstehen wir die einfachste Art der auditiven Superposition verschiedener Audiodatenströme. Diese werden individuell gewichtet zu einem Mischwert addiert und anschließend normalisiert. Das trivial anmutende Verfahren ist nach wie vor die Basis jeder mehrspurigen Audiobearbeitung, sei es bei der professionellen Produktion im Studio oder bei der Bearbeitung von Audiodaten am heimischen Computer.

Diesem linearen Mischen liegt die traditionelle analoge Technik zugrunde, bei der die einzelnen Tonspuren unangetastet bleiben und lediglich die Gewichtung der Spuren untereinander über individuelle Verstärkungsfaktoren verändert werden kann. Aufgrund der Vertrautheit von Musikern mit diesem Verfahren

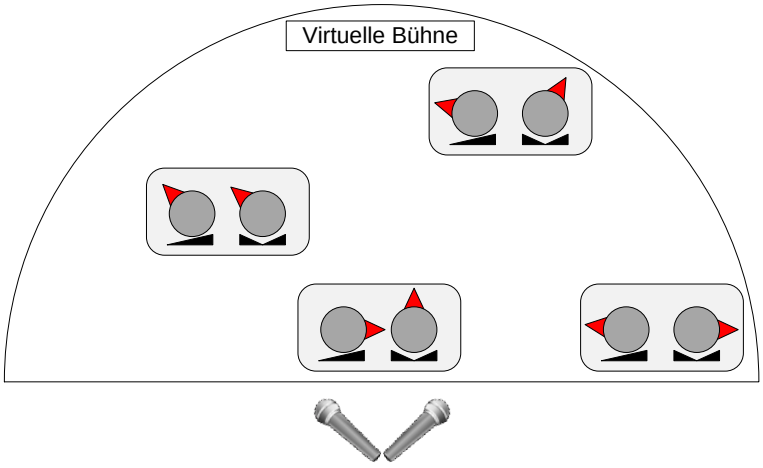

Abbildung 6.2: Positionierung von Audioquellen über Pegel und Panning

wird das lineare Mischen von unserem NMP-System unterstützt und bei den meisten Sitzungen bevorzugt verwendet.

Werden in Sitzungen pro Client zwei Audiokanäle eingesetzt, kann auch mit diesem einfachen Verfahren eine stereofonische positionsbasierte Abbildung einer virtuellen Bühne erreicht werden. Die zusammen gehörigen Kanäle werden dabei unabhängig voneinander gemischt und verlassen den Mixer wiederum als zwei eigenständige Audiokanäle.

Die Definition der Position in einer solchen stereofonischen virtuellen Bühne erfolgt dabei für jeden Client individuell: jeder Musiker muss dafür entweder zwei Audiokanäle erzeugen oder über einen Hardware Audiomixer verfügen, über den er den Pegel und die Balance (die auditive Aufteilung des Audiosignals auf zwei oder mehr Kanäle) einstellen kann. Über diese beiden Werte kann sich jeder Musiker auf einem Halbkreis virtuell frei positionieren. Den Abstand zum Mikrofon bzw. Zuhörer verstellt er über den Signalpegel – je schwächer das Signal, desto weiter entfernt erscheint die jeweilige Audioquelle. Mit der Balance (bzw. Panning) wird dagegen der Positionswinkel im Bereich zwischen ganz links und ganz rechts bestimmt. Eine gleichmäßige Verteilung des Signals auf beide Kanäle bewirkt, dass das Instrument bzw. der Sänger direkt vor dem Zuhörer wahrgenommen wird, wie in Abbildung 6.2 skizziert.

6.2.3.2 Statische Simulation virtueller Bühnen

Die Positionierung eines Musikers im virtuellen Raum über Stereofonie ist, wie oben beschrieben, einfach und für Musiker intuitiv umzusetzen. Für NMP ergibt sich daraus jedoch der Nachteil eines erhöhten Datenaufkommens. Wird eine Audiospur künstlich auf zwei oder mehrere Kanäle aufgeteilt, müssen diese individuell und damit mit multiplikativem Aufwand bearbeitet werden. Problematisch ist dabei weniger die erhöhte Belastung der CPU beim Mischen, sondern das damit einhergehende höhere Datenaufkommen, das es über das Netz zu transportieren gilt. Wenn, wie wir in späteren Abschnitten sehen werden, die Kapazität des Netzes ein limitierender Faktor für den Einsatz von NMP ist, erscheint es intuitiv wenig sinnvoll, mehrere Audiokanäle zu übertragen, die sich lediglich durch den verwendeten Verstärkungsfaktor unterscheiden.

Genau diesen Missstand geht die Server-seitige Simulation virtueller Bühnen an. Das Verfahren ist dabei naheliegend, da es die traditionelle Positionierung der Musiker über Pegel und Balance realisiert. Anders als bei der oben beschriebenen Positionierung über die lokale Hardware am Client erfolgt sie hierbei virtuell im Server. Statt das Signal beim Musiker auf zwei Stereokanäle zu verteilen, versenden die Clients jeweils

nur einen Audiokanal zum Server und stellen zusätzlich die Informationen über gewünschten Pegel l_d und Balance b_{lr} zur Verfügung.

Beim Mischen werden anschließend die Abtastwerte der Monokanäle s_m jedes Clients gemäß der vorgegebenen Werte für Pegel und Balance zu den Samples s_l und s_r für linken und rechten Kanal berechnet aus

$$s_l = s_m \cdot sin(b_{lr}) \cdot l_d \qquad \text{mit} \qquad 0 \leq b_{lr} \leq \frac{\pi}{2}, \quad 0 \leq l_d \leq 1$$
$$s_r = s_m \cdot cos(b_{lr}) \cdot l_d$$

Die erzeugten Stereokanäle werden abschließend genau wie beim oben beschriebenen linearen Mischen zweier nativer Audiokanäle gemischt und der Stereomix an alle Clients gesendet.

Der Vorteil der Server-seitigen Simulation virtueller Bühnen gegenüber der Client-seitig erzeugten Stereofonie liegt beim NMP auf der Hand: gerade auf dem kritischen Netzpfad vom Client zum Server muss nur ein Audiokanal übertragen werden, womit sich die Datenrate im Vergleich zur Stereoübertragung halbiert. Als einzigen Nachteil kann hier die leicht erhöhte Rechenlast beim Server angeführt werden, die bei einer effizienten Berechnung der Werte anhand von Sinus-Tabellen vernachlässigbar ist.

Für den Musiker ist dieses Verfahren in der Praxis genau so intuitiv anwendbar wie das gewohnte Einstellen der Parameter Pegel und Balance am Hardware Mixer. Mit einer angestrebten Verzögerung unterhalb der Wahrnehmungsgrenze erfolgt das Abmischen transparent, da der Musiker das auditive Feedback vom NMP-Server praktisch genau so wahrnimmt, wie vom lokalen Hardwaremixer. Eine weitere Annäherung an die vertraute Umgebung lässt sich bei Bedarf dadurch erreichen, dass die Eigenschaften existierender Geräte genau abgebildet werden, wie es bei Software Audiomixern üblich ist. Dabei werden die Übertragungskennlinien zwischen Regel- und Wirkwert am realen Gerät (die nicht notwendigerweise linear sein müssen, wie in obiger Formel angenommen) gemessen und bei der Simulation verwendet.

6.2.3.3 Individuelles positionsbasiertes Mischen

Die beiden bisher beschriebenen Verfahren dienen zur Erzeugung eines gemeinsamen stereofonischen Höreindrucks für eine definierte Position auf der Bühne. Üblicherweise wird dieser Ort im Mittelpunkt des Halbkreises angenommen, der Ort, an dem das Mikrofon auf der Bühne platziert wird. Diese Annahme wird in dieser Form auch bei der professionellen Produktion von Musik verwendet, da der Zuhörer genau an dieser Position platziert wird.

Mit einer nur geringfügigen Erweiterung des Server-seitigen stereofonischen Mischens lässt sich für jeden Musiker ein individueller und perspektivisch korrekter Höreindruck erzeugen, der die virtuelle Position aller Musiker zueinander berücksichtigt. Dafür ist der Referenzpunkt für jeden Teilnehmer aus der Bühnenmitte zur seiner angenommenen Position zu verschieben und die übrigen Audioquellen perspektivisch zu mischen. So erhält jeder Musiker sein individuelles Audio, das er auf einer realen Bühne wahrnehmen würde.

Diese Erweiterung erfordert einen höheren Aufwand beim Server, da die CPU-Last beim Mischen linear zur Anzahl der Musiker zunimmt. Hochoptimierte 3D-Audio-Engines (wie OpenAL, EAX, etc.), die in Spielen oder Anwendungen für virtuelle Realitäten den positionskorrekten Höreindruck berechnen, sind in der Lage, bei der in Echtzeit laufenden Berechnung bis zu 100 Audioquellen zu berücksichtigen. Wir haben in unserem NMP-System eine vereinfachte Form für die Musiker individuelle Mischfunktionalität integriert und nur eine geringe Beeinträchtigung der Skalierbarkeit durch eine höhere Belastung des Servers gemessen.

In unserer Implementierung werden die Positionen der Musiker auf der Bühne bei der Konfiguration festgelegt und für den Verlauf einer Sitzung beibehalten. Darüber hinaus wird die Orientierung aller Musiker

nach vorne senkrecht zur Bühne angenommen, welches den Gegebenheiten auf realen Bühnen entspricht. Diese Parameter können bei Bedarf aktualisiert werden, so dass die Bewegung als solche im Höreindruck wahrgenommen werden kann.

Bei der Verwendung genauer und schneller Sensoren für Orts- und Orientierungsbestimmung können die Parameter vom Client zum Sender in Echtzeit übertragen werden und so dynamisches 3D-Audio erzeugt werden. Die Umsetzung einer solchen Sensorik stand nicht in unserem Fokus, wiewohl wir Bewegungen der Musiker durch manuelle Eingaben in unserem System simulieren können.

6.2.3.4 Praktikabilität der Mischverfahren

Die Stereofonie erlaubt bereits mit zwei Audiokanälen die perspektivische Wahrnehmung einer virtuellen Bühne. Sie ist auch heute noch die meist verbreitete Methode bei der Produktion im Musikstudio. In NMP können wir Client oder Server-seitig Stereoton erzeugen. Dabei ist die Server-seitige Variante zu bevorzugen, da sie die Menge der zu übertragenden Audiodaten nahezu halbiert und für den Anwender praktisch nicht unterscheidbar vom Client-seitigen Stereo verwendet werden kann.

Mit dem Grundwerkzeug der Stereofonie für einen Referenzpunkt kann das Zuhörer relative Verfahren auf ein Musiker individuelles erweitert werden. Der dafür erforderliche Mehraufwand hinsichtlich CPU-Last beim NMP-Server beeinträchtigt seine Leistungsfähigkeit nur unwesentlich.

Allerdings haben wir während unserer Arbeit mit Musikern erfahren, dass solche – für den passiven Zuhörer höchst interessanten Erweiterungen – für den praktischen Einsatz beim aktiven Musizieren als ungeeignet erachtet werden. Tatsächlich verzichten die meisten Musiker beim Spielen meist komplett auf den räumlichen Höreindruck und verwenden Monofonie. Das scheint darin begründet zu sein, dass der Großteil der Musikinstrumente einkanalig abgegriffen wird und mit dieser Gewohnheit das Spielen auf einer virtuellen Bühne als schwer durchführbar eingestuft wird.

Der Zuhörer möchte dagegen den gewohnten Stereoton samt auditiver Lokalisation der Musiker wahrnehmen. Zur Unterstützung der Präferenzen von Musiker und Publikum haben wir in den Komponenten von NMP verschiedene Modi für das Mischen implementiert:

Client

Der Client kann Hardware-seitig so konfiguriert werden, dass er entweder ein Mono- oder ein Stereosignal abgreift. Ein Monosignal kann entweder direkt oder nach einer positionskorrekten Erweiterung auf zwei Kanäle zum Server versendet werden. Analog wird ein eingelesener Stereokanal nativ oder nach einem Downmix in Mono zum Server übertragen. Die beiden Parameter Pegel und Panorama, die zur Positionierung des Clients auf der virtuellen Bühne benötigt werden, werden global in der Konfiguration einer Sitzung definiert und können anschließend während dieser mit entsprechenden Nachrichten dynamisch angepasst werden. Den vom Server empfangenen Stereomix kann jeder Client entweder direkt als 3D-Audio ausgeben oder über ein Filter nach mono wandeln.

Server

Der Server kann Monospuren ohne und mit Positionsinformationen oder mehrstimmige Stereospuren verarbeiten. Anhand der Positionsdaten werden diese als Tonquellen auf einer virtuellen Bühne verteilt. Jeder Client hat auf dieser Bühne eine individuelle Hörposition und -Orientierung, die im Zentrum der Bühne liegend und nach vorne gerichtet den Höreindruck des Publikums entsprechen. Der Stereomix wird anschließend an alle Clients übermittelt. An Teilnehmer, die keine Positionsinformationen wünschen, wird der Mix vor dem Senden nach mono konvertiert.

Diese Funktionalität wird realisiert über zwei Filtermodule, die eine Monospur zusammen mit den Parametern Pegel und Balance stereofonisch bzw. eine Stereospur nach mono umwandelt. Bei der Umwandlung

einstimmiger Stereokanäle können die Positionsparameter extrahiert werden, andernfalls gehen diese Informationen verloren. Mehrere Instanzen dieser Module können hintereinander geschaltet werden. Mit dieser Flexibilität lässt sich das Mischen der Audiospuren an unterschiedliche Szenarien anpassen.

In einem typischen Aufbau, in dem die Clients von privaten Netzzugängen mit einem breitbandig angebundenen Server kommunizieren, ist der Flaschenhals die verminderte Datenrate der Clients in Senderichtung. Sinnvollerweise werden die Daten von den Clients zum Server in mono übertragen und dort zu einem gemeinsamen Stereosignal positionskorrekt gemischt. Die höhere Kapazität in Empfangsrichtung erlaubt anschließend die Übertragung des Stereosignals zurück zu den Clients.

Unabhängig von den herrschenden Einschränkungen bei der Übertragungskapazität empfiehlt sich dieses Vorgehen als Standardverfahren für unser NMP-System, da der überwiegende Großteil von Musikinstrumenten Monoaudio liefert. Wird die Audiohardware mit einem Stereosignal gespeist, detektieren die Filter einstimmige Kanäle und konvertieren diese nach mono mit Positionsinformationen. Liegt eine Stereospur mit zwei unabhängigen Kanälen vor, kann der Benutzer unter Berücksichtigung der verfügbaren Netzkapazität eine Stereoübertragung vorsehen oder einen Downmix auf mono erzwingen.

6.2.4 Fazit

Das vollständig automatisch durchgeführte Mischen von Audiospuren erweist sich bei genauerer Untersuchung als höchst problematisch. Grund dafür ist die Dynamik über die Zeit einer Sitzung, da die maximale Aussteuerung an der Audiohardware jedes NMP-Clients nicht vorhersehbar ist. Noch weniger lässt sich der Pegel des aus allen Audiospuren generierten Mischsignals vorab abschätzen – was in der realen Welt einen Tonmeister erfordert, ist per se nicht automatisierbar.

Zentrales Problem bei digitalem Audio ist die Pegelkontrolle und die Vermeidung der Übersteuerung beim additiven Mischen. In der analogen Welt wird dieses einfach dadurch vermieden, dass die Pegel der Einzelspuren und des Mischsignals so geregelt werden, dass diese im sicheren Sollbereich liegen. Der beschriebene Vorgang lässt sich nicht ohne Weiteres in die digitale Domäne übertragen, da hier eine Pegelanpassung immer mit dem Verlust von Dynamik einhergeht, weil der diskrete Wertebereich bei einer Verringerung der Aussteuerung eingeschränkt wird. Eine sichere Methode zur Vermeidung von Clipping beim additiven Mischen von n_C Audiospuren ist die Abschwächung aller Einzelspuren um den Faktor n_C^{-1}. Diese Sicherheit geht auf Kosten einer merklich reduzierten Dynamik einher. Eine Dämpfung der Spuren um das geometrische Mittel $n_C^{-1/2}$ berücksichtigt die verminderte Wahrscheinlichkeit, dass lokale Extremwerte für alle Clients im selben Samplewert zusammenfallen. Sie schwächt dadurch diesen negativen Effekt etwas ab auf Kosten eines Restrisikos für ein Übersteuern des Mischsignals. Zuletzt muss berücksichtigt werden, dass die mittlere Lautstärke jeder Einzelspur nur in lauten Passagen den Maximalwert erreicht und meist im mittleren Bereich liegt. Eine vorauseilende statische Pegelminderung kann bei bestimmten Fällen unnötig und störend sein.

Die Alternative zur statischen Anpassung der Aussteuerung ist die kontinuierliche automatische Pegelkontrolle (AGC), die aus naheliegenden Gründen im Musikbereich unerwünscht ist. Wir haben in NMP eine Mischform für die Anpassung gewählt, die eine fortlaufende Analyse des Pegels und eine unidirektionale graduelle Abschwächung des Mischsignals auf vordefinierte diskrete Stufen vornimmt. Da eine Abschwächung nicht revidiert wird, ist die Anzahl möglicher Anpassungen während einer Sitzung beschränkt. Die negativen Einflüsse der Vorgänge haben wir dadurch minimiert, dass der Übergang von einer Aussteuerungsstufe zur Nächsten kontinuierlich erfolgt. Dafür haben wir eine parametrisierbare Übergangsfunktion als Superposition eines linearen und eines sigmoiden Verlaufs entworfen, mit denen die Anpassung allmählich und für den Musiker nahezu unhörbar realisiert werden kann.

Das Mischen der Spuren berücksichtigt Positionsinformationen der Musiker und des Publikums auf ei-

ner virtuellen Bühne über Stereofonie. Unterstützt werden am Eingang des Mixers sowohl native Stereokanäle als auch Monokanäle mit zugehörigen Positionsangaben über Pegel und Balance. Jeder Client kann dabei definieren, wo er auf der Bühne als Audioquelle steht, und wie seine Hörposition und -richtung sind. Diese Trennung der aktiven und passiven Rolle erlaubt vollkommene Flexibilität bei der Gestaltung des virtuellen Raums, darunter auch den meist verwendeten Aufbau, dass jeder Musiker zwar von seiner individuellen Position spielt, jedoch den Höreindruck des vor der Bühne sitzenden Publikums wahrnimmt. Eine dynamische Anpassung des Positionsparameter wurde betrachtet, jedoch für NMP aufgrund mangelnder Relevanz nicht weiter verfolgt.

Bei der Untersuchung wurden darüber hinaus NMP-typische Belange hinsichtlich Netz- und CPU-Last weitestgehend berücksichtigt. So wird ein von der Clienthardware eingelesener Stereokanal auf Einstimmigkeit geprüft und nach Möglichkeit in einem Monokanal mit Positionsinformationen gewandelt, womit praktisch die Halbierung der ausgehenden Netzlast dieses Clients mit vernachlässigbar geringem CPU-Aufwand erreicht wird. Auf der Serverseite ist der Mixer die zentrale Komponente und wurde so gestaltet, dass der relevante Faktor CPU-Last geschont wird. Bei der Auswahl der Algorithmen wurde auf Einfachheit und schnelle Abarbeitung hin optimiert. So basieren alle Berechnungen auf Integer-Arithmetik, häufig verwendete Werte (wie die sigmoide Übergangsfunktion) werden vorberechnet und anschließend aus Tabellen ausgelesen.

6.3 Fehlerverdeckung

Das Konzept von NMP, einen Betrieb über eine Abwägung zwischen den Kenngrößen Latenz und Audioqualität zu erreichen, impliziert das Auftreten von Paketverlusten in Form von beim Transport tatsächlich verworfenen oder zum Verarbeitungszeitpunkt verspätet eintreffenden Daten. Aufgrund des Best-Effort Charakters des Internets sind solche Paketverluste nicht zu vermeiden und stellen für alle interaktiven Anwendungen, bei denen die Neuübertragung der betreffenden Daten nicht möglich ist, eine große Herausforderung dar.

In Audioanwendungen führen Paketverluste zu Lücken im Datenstrom, deren Auftreten sich auf die wahrgenommene Audioqualität besonders unangenehm auswirkt. Eine Diskontinuität im Zeitbereich erstreckt sich bei der Transformation in den Frequenzbereich über das gesamte Spektrum. Dem in diesem Modus arbeitenden menschlichen Hörapparat fallen solche Fehlstellen daher zwangsläufig auf und machen sich dabei sehr störend bemerkbar. In allen Audioanwendungen, die mit potentiellen Paketverlusten konfrontiert sind, müssen daher geeignete Maßnahmen zur Erkennung von Lücken im Datenstrom und zur Korrektur oder Abschwächung ihres auf die subjektiv wahrgenommene Audioqualität negativen Einflusses getroffen werden.

In diesem Abschnitt wollen wir auf dieses Kernproblem für NMP eingehen.

6.3.1 Behandlung von Paketverlusten in Audioanwendungen

Für die Behandlung von Paketverlusten stehen prinzipiell zwei zu unterscheidende Ansätze bereit: proaktive Verfahren basieren darauf, dass der Sender die Nutzdaten um redundante Informationen ergänzt, die bei Datenverlusten bis zu einem definierten Grad eine Rekonstruktion der ursprünglichen Daten ermöglichen. Reaktive Verfahren werden dagegen beim Empfänger angewandt, um auftretende Lücken ohne zusätzliche Informationen zu schließen. Die Möglichkeit zur Rekonstruktion sind dabei stark eingeschränkt, vielmehr wird dabei versucht, die Fehlstellen bestmöglich zu verdecken.

6.3.1.1 Senderseitige Vorwärtsfehlerkorrektur

Der proaktive Ansatz ist nicht auf den Audiobereich beschränkt, sondern kommt als universelles Verfahren immer dann zum Einsatz, wenn in einer Umgebung mit bekannter und relativ konstanter Datenverlustwahrscheinlichkeit verfälschte Daten wiederhergestellt werden müssen. Entsprechende Verfahren zur Vorwärtsfehlerkorrektur (FEC, forward error correction) kommen in den unterschiedlichsten Anwendungen vor, aus der Informationstechnologie lassen sich exemplarisch die folgenden aufführen:

Kommunikationssysteme

Erfolgt die Datenübertragung über ein störanfälliges Medium wie bei der drahtlosen Kommunikation über die Luft, sind Übermittlungsfehler unvermeidbar. Um über solche Medien auf Anwenderebene eine praktikable Kommunikation zu ermöglichen, werden bereits auf der Sicherungsschicht Verfahren implementiert, die kontinuierlich auftretende Fehler erkennen und korrigieren. Als bekannteste Beispiele seien das alltäglich verwendete WLAN nach IEEE802.11 [40] oder die redundanten Codecs beim Mobilfunk zu nennen. Daneben wird auch unser aller Zugang zum Internet über ADSL [3] durch Vorwärtsfehlerkorrektur abgesichert.

Speichermedien

Ohne geeignete Maßnahmen wäre eine Audio- oder Daten-CD nach kürzester Zeit nicht mehr zu gebrauchen. Durch mechanische Beanspruchung bei der Bedienung und beim Datenzugriff entstehen zwangsläufig Kratzer auf dem Medium, die beim weiteren Auslesen zu Bitfehlern führen würden. Die im 'YellowBook' des ISO-9660 Standards spezifizierte Vorwärtsfehlerkorrektur erfordert für die Sicherung eines 2048 Byte Blocks 288 Bytes für FEC [42]. Nur so können die optischen Datenträger ihre Langlebigkeit und Robustheit bereitstellen, die ihren Siegeszug begründet hat.

Integrierte Schaltungen

Integrierte Schaltkreise müssen bei Einsätzen in sicherheitskritischen Bereichen technisch nicht vermeidbare Bitfehler, die durch Störungen oder Strahlung verursacht werden, erkennen und korrigieren können. Dabei kommen in Hardware umgesetzte Fehlerkorrekturverfahren zum Einsatz, beispielsweise in Form von ECC RAM als flüchtigen Speicher, der Ein-Bit-Fehler korrigieren und Zwei-Bit-Fehler zu erkennen vermag. In der Raumfahrt gelten aufgrund der starken kosmischen Strahlung noch höhere Anforderung an der Korrekturfähigkeit, kritische Berechnungen werden bisweilen von mehreren redundanten Systemen parallel berechnet und per Mehrheitsentscheid getroffen.

Allen Verfahren gemein ist die Ergänzung der Nutzdaten um Redundanzen, die sich nach dem Grad und dem Muster der erwarteten Fehlerwahrscheinlichkeit und der erforderlichen Korrigierbarkeit richten. Während einzelne Bitfehler durch ein einzelnes Paritätsbit pro Datenwort erkennbar sind, erfordert die Korrektur verfälschter Bits die Anwendung von Fehlerkorrekturcodes mit deutlich höherem Datenaufkommen.

In Szenarien, denen ein burstartiges Fehlermuster zugrunde liegt, ist neben der lokalen Absicherung auch eine zeitliche Entkoppelung des Fehlerrisikos erforderlich. So böte eine Absicherung eines einzelnen Blocks bei einer CD keinen Schutz gegen Kratzer, die meist lokale Fehler über einen gesamten Block verursachen. Diesem Risiko begegnet man dadurch, dass bei der Absicherung verschiedene Blöcke verwürfelt werden, so dass bei einer lokalen Beschädigung des Mediums ein zerstörter Block aus den weitflächig verteilten Sicherungsinformationen rekonstruiert werden kann. Ähnlich funktionieren die Verwürfelungsmechanismen beim Interleaving von DSL, die das Datenverlustrisiko bei kurzzeitigen Unterbrüchen verringern.

Für den Einsatz in NMP sind Vorwärtsfehlerkorrekturen wenig geeignet: wie der Name bereits impliziert, ist ihre Anwendung mit einer algorithmischen Verzögerung verbunden, die sich aus der erforderlichen Pufferung ergibt. Wählt man für die FEC die System weite Blockgröße von 128 Samples, fällt zwar keine zusätzliche Pufferlatenz an. Allerdings macht eine lokale Absicherung isolierter Audiopakete auf der Anwendungsschicht

des IP Protokollstapels keinen Sinn, da Übertragungsfehler bereits in den unteren Schichten geprüft und fehlerhafte Pakete verworfen werden.

Auf dieser Ebene ermöglicht die Verwendung von FEC mit Blockgrößen über mehrere Audiopakete die Rekonstruktion einzelner Paketverluste, so wie es innerhalb des *Multicast Backbone* (MBone [10]) für Internettelefonie und -Konferenzen Verwendung findet. Dabei werden n Audiopakete durch ein zusätzliches Redundanzpaket zu einer $n+1$ Gruppe erweitert, welches die Rekonstruktion eines Paketverlustes innerhalb der jeweiligen Paketgruppe ermöglicht. Ein entsprechendes FEC-Schema wird zur allgemeinen Verwendung bei der Datenübertragung unter [78] vorgeschlagen.

Für NMP haben wir nach der Erprobung verschiedener FEC Verfahren von einem Einsatz abgesehen, da das knappe Latenzbudget einer zusätzlichen Verzögerung grundlegend gegenübersteht. Auch in Szenarien, in denen entsprechende Maßnahmen einsetzbar wären, ist der potentielle Nutzen gering. Dazu stelle man sich vor, das Versenden der Audiopakete könnte um eine Pufferlänge verzögert werden. Über die zwei im Sendepuffer verfügbaren Pakete könnte dann eine (3,2)-FEC Sicherung angewandt werden, die den beiden Paketen ein zusätzliches Redundanzpaket zur Seite stellt. Wir erhöhen damit also die Datenrate um 50%, um einen Paketverlust über zwei Pakete rekonstruieren zu können. Dieser Mehraufwand steht in keinem Verhältnis zu der Anzahl potentieller Rekonstruktionen, die in einem für den Einsatz von NMP geeigneten Netz zu erwarten wären. Darüber hinaus kann die höhere Datenrate die Paketfehlerrate dadurch beeinflussen, dass die höhere Netzlast die Gefahr von Stauungen und Paketverwürfen auf den Netzknoten erhöht.

Wir verzichten in unserem NMP-System daher vollständig auf FEC Maßnahmen und stellen im Folgenden unsere Verfahren für die effiziente Verdeckung von Paketverlusten vor.

6.3.1.2 Empfängerseitige Fehlerverdeckung

Ohne die Verwendung von FEC obliegt es dem Empfänger, Datenlücken durch geeignete Maßnahmen zu schließen. In Audioanwendungen geht es dann konkret darum, welche Daten anstelle eines Paketverlustes über die Soundkarte auszugeben sind. Im einfachsten Fall bietet es sich an, ein Paket akustischer Stille, also eines vollständig mit Nullen gefüllten Datenblock auszugeben. Dieser naheliegende Ansatz ist am einfachsten zu realisieren und wurde bereits zu der Anfangszeit der IP Audiokommunikation in [49] diskutiert. In der Literatur wird das Verfahren als *Silence Substitution* geführt. In weiteren Untersuchungen, wie bspw. in [36], wurde es allerdings für den Einsatz in IP Telefonieanwendungen als wenig effizient bewertet. Problematisch ist dabei, dass das Einfügen eines Null-Blocks eine abrupte Diskontinuität in den Signalverlauf verursacht. Typischerweise manifestiert sich eine solche Modifikation in einem Knacksen oder Scheppern der Audioausgabe, verursacht durch die sprunghafte Hubänderung der Lautsprechermembrane.

Eine abgewandelte Form der Fehlerverdeckung ist die *Noise Substitution*, also der Ersatz der Datenlücke durch ein Rauschsignal. In Hörtests wie die in [93] wird dieser Ansatz von den Testpersonen meist besser beurteilt als die Silence Substitution. Dies wird auf frühere Untersuchungen in [58] zurückgeführt, die die Fähigkeit des menschliche Hörapparates nachweisen, von Störgeräuschen überlagerte Sprache verarbeiten zu können. Aufgrund der ähnlich einfachen Umsetzbarkeit wird die Noise Substitution daher als Ersatz für die Silence Substitution vorgeschlagen und hat so Einzug in Telefoniecodecs gehalten, wo sie Bestandteil der Empfehlungen für Audio-Konferenzen über RTP [41] ist.

Ein weiteres etabliertes Ersetzungsverfahren ist die Blockwiederholung bzw. *Repetition*, bei der bei einem Paketverlust der zuletzt gespielte Block wiederholt wird. Bei neueren Audiokarten wird dieser Ansatz direkt in der Hardware umgesetzt, indem der zuletzt ausgebende Block so lange wiederholt wird, bis der nächste verfügbar ist. Der plausible Vorteil einer Repetition liegt darin, mit einer erhöhten Wahrscheinlichkeit von einer Stationarität im Signalverlauf profitieren zu können: im Falle einer Sprachkommunikation steigt die

Wahrscheinlichkeit mit abnehmender Blockgröße, dass zwei benachbarte Blöcke Teil eines selben Phonems oder einer Sprechpause sind, während dies bei Musikanwendungen entsprechend für Noten und Klänge gilt. Kombiniert mit einer überlagerten kontinuierlichen Dämpfung des wiederholten Datenblocks ist die Repetition in [1] als Teil der Spezifikationen des GSM Mobilfunkstandards enthalten.

Diese und ähnliche Ersetzungsverfahren, die einem festen Schema folgend mit geringer Komplexität anwendbar sind, haben die gemeinsame Schwäche der abrupten Diskontinuitäten an den Rändern. Sie ersetzten die Datenlücken mit vorhandenen oder generierten Daten, ohne auf den Kurvenverlauf des Audiosignals vor und nach der Lücke einzugehen.

Eben diesem Problem widmet sich die Klasse interpolativer Verfahren, die auf Analysen der dem Paketverlust benachbarten Daten beruht. In [32] wird ein Basisverfahren vorgestellt, welches die Frequenzen und Lautstärken an den beiden Seiten des Paketverlustes schätzt und zwischen diesen in der Lücke interpoliert. Eine Erweiterung des Verfahrens unter [94] schlägt statt einer Interpolation die Erkennung periodischer Verläufe und deren Repetition innerhalb der Lücke vor. Weiterhin differenziert der Ansatz zwischen Sprache und Sprechpausen und stellt ein Schema vor, dass eine fallbezogene Fehlerverdeckung in Form von Interpolation oder Repetition ermöglicht. Als letztes Beispiel eines interpolativen Ansatzes ist das in [81] vorgestellte Verfahren zu nennen, welches eine Überbrückung der Datenlücke durch die zeitliche Dehnung der benachbarten Blöcke zu erreichen versucht.

In vielen Sprachcodecs ist die Rekonstruktion von Lücken im Audiodatenstrom integraler Bestandteil des Decoders. Ein bekanntes Beispiel ist der ITU Sprachcodec G.723.1 [44], dessen Encoder auf linearen Vorhersagekoeffizienten (LPC, linear prediction coefficient) basiert. Paketverluste werden verborgen, indem zwischen den LPC des vorhergehenden und nachfolgenden Paketes interpoliert wird und so die wesentlichen Merkmale des Audiosignals rekonstruiert werden. Auch hierbei erfolgt eine Unterscheidung zwischen Sprache und Sprechpausen, so dass kurze Datenlücken für das subjektive Hörempfinden effizient verdeckt werden.

Die bisher aufgeführten Verfahren ersetzen die Fehlstellen vollständig mit den berechneten Daten und behalten dabei die Anzahl der Abtastwerte bei. Das so genannte *Splicing* verdeckt Paketverluste, indem die Fehlstelle ignoriert und der Audiodatenstrom um die Lücke gekürzt wird – das nächste vorhandene Paket wird unmittelbar an das letzte 'geklebt'. Laut Untersuchungen in [33] ist das Verfahren nur bei sehr kurzen Lücken unter 4 ms und geringer Paketverlustrate unter 3 % einsetzbar. Zudem verändert sich das Timing des Audiosignals, die Abspieldauer wird mit jeder durchgeführten Operation immer kürzer als die Aufnahmedauer. Für synchrone Anwendungen ist es somit nicht geeignet, da es beim Empfänger unweigerlich zu Puffer-Unterläufen kommt.

Die Komplexität der beschriebenen Verfahren steigt in der aufgelisteten Reihenfolge kontinuierlich. Die statische Substitution von Paketverlusten über *Silence Substitution* ist praktisch aufwandslos zu erreichen, während für die *Noise Substitution* eine Pegelanpassung des Rauschens und damit eine Messung der Lautstärke links und rechts der Lücke erforderlich ist. Gleiches gilt für die Repetition, die ohne Pegelanpassung keinen Zusatzaufwand verursacht. Interpolative Verfahren können bei der Analyse beliebig komplex werden, wenn Sprünge an den Rändern vermieden werden sollen und gleichzeitig der Signalverlauf hinsichtlich Lautstärke und Tonhöhe ohne hörbare Diskontinuitäten zu rekonstruieren ist.

Unser NMP-System unterscheidet sich grundlegend von den Übertragungssystemen für Sprache, die als Zielanwendung für den Entwurf bzw. Untersuchung der vorgestellten Verfahren dienen. Ganz augenscheinlich hat eine Musikanwendung komplett andere Charakteristika, die Anforderungen und Rahmenbedingungen beim Entwurf eines geeigneten Verfahrens beeinflussen: zu nennen sind da an erster Stelle die im Vergleich zur Sprache erweiterte Dynamik und der breitere Frequenzbereich. Die Komposition multipler Instrumente bzw. Tonspuren zu einem Mischsignal führt daneben zu einer höheren Kontinuität – ein Äquivalent zu

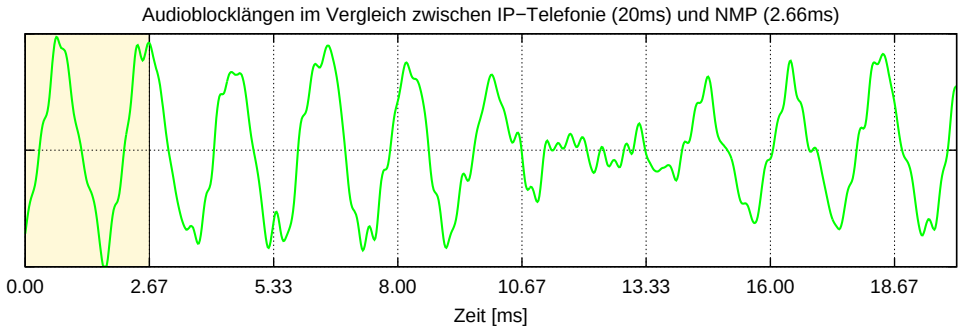

Abbildung 6.3: Blockgrößen von NMP und IP-Telefonie

'Sprechpausen' fehlt. Von technischer Seite bestimmend ist die Zusammensetzung eines NMP-Systems aus den Endsystemen Client und Server, die funktional völlig unterschiedliche Anforderung an die Fehlerverdeckung stellen. Weiterhin relevant sind die im Vergleich zur Sprachübertragung bei NMP verwendeten deutlich kürzeren Audioblöcke (2.66 ms gegenüber ca. 10-40 ms), die den Einsatz existierender Verfahren behindern.

Wie bedeutend dieser Unterschied ist, verdeutlicht die Abbildung 6.3. Dargestellt ist der Verlauf eines realen Audiosignals über die Dauer von 20 ms und damit in typischen Blockgrößen von Sprachcodecs. Das vertikale Raster ist in den bei NMP verwendeten Abständen von 2.66 ms aufgetragen. Der exemplarisch hervorgehobene erste Audiodatenblock macht sehr genau deutlich, wie unterschiedlich die vorgestellten Verfahren auf diese Datenbereiche angewendet werden können. Während das gesamte Signal mehrere vollständige Perioden der gemischten Oberwelle im Audiosignal beinhaltet, passt in den NMP-Blöcken meist nur ein vollständiges Intervall. Eine Analyse von Tonhöhe und Oberwellen, wie sie die vorgestellten Verfahren verwenden, wäre für NMP somit völlig ungeeignet, da diese Informationen kaum sinnvoll Verwendung finden können.

Aufgrund dieser Einschränkungen haben wir für NMP ein eigenes Verfahren entworfen, welches die Grundideen bestehender Ansätze aufgreift und den besonderen Anforderungen entsprechend kombiniert.

6.3.2 Anforderungen an ein Fehlerverdeckungsverfahren für NMP

Ein Blick auf die verschiedenen Betriebsmodi und die unterschiedlichen Anforderungen in den beteiligten Endsystemen offenbart, warum keines der vorgestellten Verfahren durchgängig in unserem NMP-System verwendbar ist. Ganz grundlegend unterscheiden sich die Anforderungen im Live- von denen des On-Demand-Modus: während des Musizierens liegt der Fokus bei einer möglichst effizienten Fehlerverdeckung von Paketverlusten unter Hinnahme von Qualitätseinbußen. Für denn nachgelagerten On-Demand Abruf der aufgezeichneten Audiodaten müssen dagegen für die Bereitstellung bestmöglicher Audioqualität die Daten lückenlos sein, es müssen daher Mechanismen zur vollständigen Rekonstruktion angewendet werden.

Neben dieser globalen Forderung nach Fehlerverdeckung und Rekonstruktion schränken die Betriebsparameter der beteiligten Komponenten im Live-Modus die Auswahl geeigneter Korrekturverfahren auf unterschiedlichen Arten ein. Als naheliegende Unterscheidung ist die Komplexität des gewählten Verfahrens zu nennen: auf der einen Seite haben wir den NMP-Client, der sich lediglich um seinen eigenen Audiodatenstrom kümmert und daher ein Vielfaches der Komplexität für die Fehlerkorrektur einsetzen kann als der Server, der eine Vielzahl von Audiodatenströmen gleichzeitig zu bearbeiten hat.

Darüber hinaus bedingen die unterschiedlichen Zuständigkeiten der Komponenten bei der Bearbeitung abweichende Rahmenbedingungen für das Auftreten und die Erkennung von Paketverlusten und damit Endsystem spezifischer Vorgaben. Ohne allzu weit in die Ausführungen der folgenden Abschnitte vorgreifen zu

müssen, sind einige relevante Fakten ohne umfassende Untersuchung nachvollziehbar:

- Eine zweifelsfreie Detektion eines Paketverlustes ist beim Client deutlich einfacher, da hier etwa doppelt so viele Pakete in den De-Jitter Puffern vorhanden sind als beim Server.
- Die Verarbeitung beim Client ist synchron und von der Audiokarte vorgegeben, während der Server asynchron arbeitet. Eine Erkennung von Mehrfachfehlern ist beim Server daher weitaus ungenauer. Darüber hinaus treten Paketverluste in Form verworfener Pakete zur Paketsynchronisation nur beim Server auf.
- Im Gegensatz zum Server verfügt der Client bei Paketverlusten mit seinen eigenen Audiodaten zumindest über die Kenntnis eines Teiles des Audiosignals, der zur Fehlerverdeckung herangezogen werden kann.

Da somit ein einheitliches Verfahren beim Client und Server für die Behandlung von Paketverlusten nicht umsetzbar ist, haben wir für NMP ein Verfahren entwickelt, das während des Musizierens eine asynchrone Fehlerverdeckung realisiert und die vollständige Rekonstruktion aller Audiodaten für den On-Demand Abruf bietet.

6.3.3 Nachgelagerte Korrektur für den On-Demand Einsatz

Um nach einer erfolgreichen Live-Sitzung die gesammelten Daten sinnvoll archivieren und weiter verarbeiten zu können, müssen diese vollständig und fehlerfrei sein. Die während des Musizierens tolerierten Paketverluste sind für eine weitere Verwendung der Audiodaten nicht akzeptabel und müssen wiederhergestellt werden. Da dieser Prozessschritt außerhalb der Echtzeit-Sitzung erfolgt und damit keiner Latenzschranke unterliegt, können für die Korrektur beliebige Ansätze gewählt werden, einschließlich der einleitend beschriebenen sehr komplexen Rekonstruktionsverfahren.

Da jedoch die Qualität des ursprünglichen Audiodatenstroms mit keiner – auch beliebig komplexen – Operation rekonstruiert werden kann, setzen wir auf einen Mechanismus zur Neuübertragung von im Netz verlorenen Audiopaketen. Das umgesetzte Schema arbeitet abschnittsweise und basiert auf asynchrone Anforderungen des Servers, die vom Client bearbeitet werden.

Nach der regulären Bearbeitung aller zu einem Block bzw. Abschnitt gehörenden Audiopakete stellt der Server für jeden Client eine Liste aller aus diesem Abschnitt fehlenden zusammen und übermittelt ihnen diese. Die betreffende Signalisierung erfolgt über ein der Echtzeit-Audioübertragung überlagertes Protokoll und stellt sicher, dass der Transport der Live-Audiodaten dadurch nicht behindert wird.

Der adressierte Client hat nach Erhalt der Anfrage zwei Bearbeitungsmöglichkeiten, die durch die Betriebsparameter Netz- und Speicherkapazität bestimmt werden:

Neuübertragung angeforderter Audiopakete während der Echtzeit-Sitzung

Dieser Modus kann dann verwendet werden, wenn die Netzanbindung des betreffenden Clients über ausreichend Kapazität verfügt, um die fehlenden Daten gleichzeitig neben den aktuellen zu übertragen, ohne diese zu beeinflussen. Ist diese Bedingung erfüllt, stellt unser NMP-System in dieser Betriebsart einen Mechanismus bereit, der die zu wiederholenden Pakete so in den laufenden Datentransfer einfügt, dass keine wesentlichen Änderungen der Parameter Latenz und Jitter erfolgen. Dieser Modus erfordern vom Client die Zwischenspeicherung zurückliegender Pakete, um diese bei Bedarf neu übertragen zu können. Die Größe des dafür erforderlichen Puffers wird dabei von der gewählten Abschnittsgröße und der Reaktionsdauer des Servers bestimmt. In unserem NMP-System verwenden wir als Standardwert Abschnitte von jeweils einer Sekunde mit entsprechend 375 Paketen. Neben diesem Vorrat muss der Client auch die Pakete des aktuellen Abschnittes sammeln, bis die Anfrage vom Server empfangen und bearbeitet ist. Somit wird für eine sichere Anwendung des Verfahrens beim Client ein

Puffer vom Doppelten der verwendeten Abschnittsgröße benötigt. Server-seitig gilt diese Anforderung analog, d.h., auch hier sind zwei Sätze von Puffern erforderlich – ein zurückliegender, der bei Bedarf noch vervollständigt wird, und ein aktueller.

Neuübertragung aller Pakete nach der Echtzeit-Sitzung

Verfügt ein Client nicht über ausreichend Netzkapazität, um die angeforderten Pakete während der Live-Sitzung nachzuliefern, werden die betreffenden Pakete zunächst lokal gesammelt und erst nach Beendigung der Sitzung am Stück zum Server übertragen. Bedingung für diesen Modus ist, dass beim Client ausreichend Ressourcen für die Speicherung der Pakete vorhanden sein müssen, sei es im Hauptspeicher oder auf Festplatte. Da hierbei der abschließende Block-Transfer aller gesammelten Daten außerhalb der Sitzung stattfindet, ist kein dedizierter Mechanismus zum Paket-Scheduling notwendig, die Daten können über eine TCP-Verbindung direkt übertragen werden.

Eine Extremform des ersten Modus ist die Wahl einer unendlichen Abschnittsgröße, wobei der Client alle während einer Sitzung zum Server gesendeten Daten lokal vorhält.

Unabhängig vom gewählten Modus versetzt dieses Verfahren den NMP-Server in die Lage, die währen einer Sitzung aufgetretenen Paketverluste zu vervollständigen und alle ausgetauschten Audiodaten zu archivieren. Eine nachträgliche Bearbeitung und Verwertung der Daten kann ohne Qualitätsverluste erfolgen, d.h., eine erfolgreich durchgeführte Sitzung kann anschließend als Eingangsmaterial im Studio verwendet werden.

6.3.4 Umgang mit Paketverlusten zur Laufzeit

Die Fähigkeit, alle Audiodatenströme einer Sitzung vollständig archivieren zu können, ist ein Feature von NMP, welches erst dann relevant wird, wenn die Behandlung von Paketverlusten während des Musizierens ebenso effektiv erfolgt. Denn nur wenn die durch Datenlücken verursachten Störungen erfolgreich gemildert werden, können Sitzungen sinnvoll durchgeführt werden.

Im Rahmen einer an unserem Institut erstellten Diplomarbeit wurde in [11] die Anwendbarkeit der einleitend vorgestellten etablierten Verfahren für NMP untersucht. Eine direkte Übernahme existierender Ergebnisse war dabei nicht unmittelbar möglich, wofür im Wesentlichen die in Abschnitt 6.3.1.2 beschriebenen Unterschiede beim verwendeten Audioformat und bei den Verarbeitungsparametern ausschlaggebend sind.

6.3.4.1 Bestimmung des Basisverfahrens

Die Effektivität unterschiedlicher Ansätze für die Verwendung in NMP wurde daher explizit evaluiert, wobei diese Untersuchungen durch das Fehlen objektiver Bewertungskriterien erschwert wurden. Der für Qualitätsmessungen oft herangezogene Störabstand (PSNR) stellte sich reproduzierbar als unbrauchbar heraus, da er mit subjektiven Bewertungen nur wenig korreliert. Modell basierte Qualitätsmaße wie PEAQ sind für die Bewertung minimaler Abweichungen konzipiert und versagen bei disruptiven Fehlern, wie sie von Paketverlusten verursacht werden. Auf diese Problematik gehen wir bei der Diskussion von geeigneten Audiokompressionsverfahren in Abschnitt 6.4 genauer ein.

Die Durchführung der Untersuchungen und die gewonnenen Erkenntnisse sind aufgrund dieser Gegebenheit immer subjektiver Natur und repräsentieren die gemittelte Bewertung der Projektbeteiligten. Vorbehaltlich des subjektiven Charakters decken sich unsere dabei gewonnenen Resultate mit existierenden Ergebnissen, die sich für den Entwurf des Fehlerverdeckungsverfahrens für NMP wie folgt zusammenfassen lassen:

1. *Silence Substitution* ist auch bei geringer Paketverlustrate wenig akzeptabel, da die resultierenden Diskontinuitäten deutlich wahrnehmbares und unangenehmes Knacken verursachen.

2. Bei Verwendung von *Noise Substitution* mit einem im Vergleich zur Umgebung um 3 dB gedämpften Pegel (entspricht der halben mittleren Intensität des Audiosignals in den benachbarten Paketen) wird das Knacken abgeschwächt, das vorhandene Rauschen wird als wenig störend wahrgenommen.
3. Eine zusätzliche Glättung an den Rändern verbessert die Anwendung der *Noise Substitution* dahin gehend, dass die Knackser nahezu komplett verschwinden. Der verbleibende störende Faktor des Rauschens lässt sich dagegen prinzipbedingt nicht vermeiden.

Auf Grundlage dieser Erkenntnisse haben wir für den Einsatz in Live-Sitzungen in NMP die *Noise Substitution* als erstes Basisverfahren für die Fehlerverdeckung gewählt. Es eignet sich für den Einsatz in lokalen Netzen bei Paketverlustraten von unter einem Prozent sehr gut, da die damit verdeckten Datenlücken kaum wahrnehmbar sind. Die verbleibenden Störungen sind angesichts der geringen Komplexität des Verfahrens in solchen Szenarien tolerierbar.

Mit steigender Paketverlustrate machen sich diese Auswirkungen jedoch immer stärker störend bemerkbar, insbesondere bei Mehrfachfehlern, die zu einem länger andauernden Rauschen korrigiert werden. Bei Verwendung hochwertiger Lautsprecher oder Kopfhörer fällt so die subjektiv wahrgenommene Audioqualität bereits ab einer mittleren Fehlerrate über 1% merklich ab.

In NMP-Szenarien im Internet sind solche burstartigen Mehrfachfehler mitunter auch bei geringeren Verlustwahrscheinlichkeiten zu erwarten, da Stauungen in Routern meist dazu führen, dass mehrere aufeinanderfolgende Pakete verzögert eintreffen und verworfen werden. Empirisch haben wir dabei einen Schranke von etwa 1.5% ermittelt, ab der die geglättete *Noise Substitution* keine zufriedenstellende Fehlerverdeckung mehr bietet.

Für den Einsatz in solchen Umgebungen haben wir ein Verfahren entworfen, welches auf den vorgestellten Ansätzen in [94, 81] beruht und darauf abzielt, Datenlücken so zu ersetzen, dass der Verlauf des Audiosignals zu den Nachbarblöcken ansatzlos verläuft, bei gleichzeitiger Beibehaltung der Signalcharakteristik in unmittelbarer Umgebung.

Die Bestimmung eines geeigneten Signalverlaufs für den Ersatz eines Paketverlustes verläuft analytisch über die zurückliegenden Audiopakete über mehrere iterative Durchläufe einer Minimalsuche. Dabei wird ein Fenster mit der initialen Breite des Audiodatenblocks über den bisherigen Signalverlauf nach links geschoben. An jeder Sampleposition wird geprüft, ob der betrachtete Signalabschnitt als Ersatz in Frage kommt. Die Kriterien dafür sind dabei:

- minimale Abweichung der Abtastwerte zu den Rändern
- minimale Abweichung der Gradienten an den Rändern

Ein idealer Abschnitt zeichnet sich dadurch aus, dass die Übergänge zwischen dem eingefügten Signal und der direkten Nachbarschaft kontinuierlich verlaufen, d.h., es besteht keine Diskontinuität zwischen den Samplewerten und die Steigung bleibt konstant. Während das Verfahren für periodische Signalverläufe immer einen idealen Abschnitt findet, ist dies mit natürlichen Audiodaten eher unwahrscheinlich. Daher versucht unser Verfahren, die Suche über eine zusätzliche Dehnung bzw. Stauchung des Signals zu erweitern. Dazu wird die Suche mit einem in der Breite variierenden Suchfenster mehrmals wiederholt. Aus den dabei ermittelten Bewertungen anhand obiger Kriterien wird dann der am besten geeignete Abschnitt auf die Blockbreite skaliert und an Stelle des Paketverlustes eingefügt. In Anlehnung an die verwendeten Komponenten aus bestehenden Ansätzen haben wir diesen Ansatz *Scaled Pattern Substitution* benannt, die wir in diesem Kapitel mit *SPS* abkürzen werden.

Bleibt die Suche innerhalb des Suchparameterraumes erfolglos, wird für die betreffende Datenlücke auf die *Noise Substitution* zurückgegriffen.

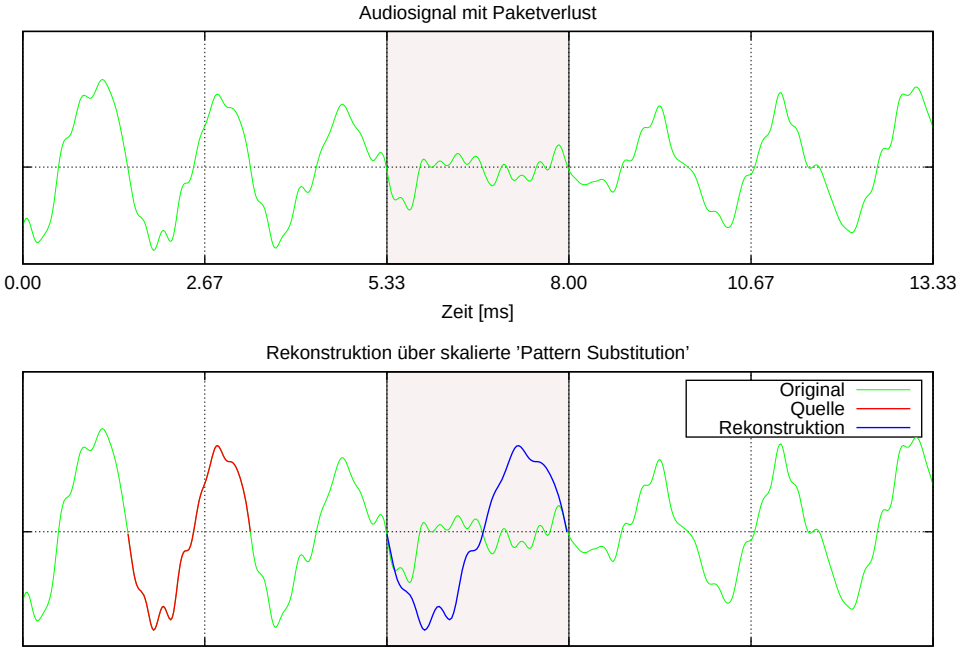

Abbildung 6.4: Fehlerverdeckung über Scaled Pattern Substitution

In Abbildung 6.4 ist das Ergebnis einer solchen Suche anhand einer Lücke in einem Signalverlauf realer Audiodaten dargestellt. In der oberen Hälfte ist das originale Audiosignal aufgetragen, bei dem die graue Fläche den zu korrigierenden Paketverlust entspricht. Das vertikale Gitternetz stellt dabei die Paketgrenzen dar. Zur besseren Verdeutlichung haben wir dabei einen Abschnitt gewählt, der sich von seiner homogenen Umgebung abhebt.

In der unteren Hälfte ist der Signalverlauf nach der Korrektur abgebildet. Das Ergebnis der Suche ist der rot markierte Abschnitt, der innerhalb der zwei dem Paketverlust vorhergehenden Abschnitte ermittelt wurde. Dieser ist knapp 20% schmaler als das zu ersetzende Teilstück, der nach einer Skalierung als blau gefärbte Rekonstruktion die Lücke abdeckt.

Wir sehen, dass die Fehlerverdeckung mit nur minimalen Sprüngen bei kontinuierlichem Signalverlauf gelungen ist. Rein visuell erkennen wir jedoch, dass das rekonstruierte Signal stark vom ursprünglichen abweicht. Indem es im Wesentlichen die lokale Signalcharakteristik beibehält, wird der abrupte und in diesem Beispiel nur einmalig auftretende Wechsel des originalen Signalverlaufs unterschlagen. Dieser Einschränkung unterliegt jedes Fehlerverdeckungsverfahren, da in dem Fall, dass die verlorenen Daten grundlegend andere Charakteristiken haben als die unmittelbare Umgebung, keine Rekonstruktion möglich ist.

Im Ergebnis werden mit diesem Verfahren bei NMP jedoch sehr gute Ergebnisse erreicht. Paketverluste sind auch hierbei ab etwa einer Paketverlustrate von 1% wahrnehmbar, jedoch im Gegensatz zur *Noise Substitution* deutlich weniger störend. Hinsichtlich der Komplexität ist diese *SPS* deutlich anspruchsvoller, da die Suche nach geeigneten Ersatzabschnitten und deren Skalierung einen erheblichen Rechenaufwand erfordert.

Besonders für den Server ist bei der gleichzeitigen Bearbeitung mehrerer Sitzungen und somit vieler paralleler Audiodatenströme eine sichere Anwendung dieses Verfahrens problematisch, da es bei einem gleichzeitigen Auftreten von Paketverlusten bei mehreren Audiodatenströmen die für einem Intervall verfügbare Bearbeitungszeit überschreiten kann. Für diese Komponente haben wir eine auf Geschwindigkeit optimierte Variante entwickelt, die unter Einschränkung des Suchparameterraumes gute Ergebnisse in akzeptabler Laufzeit liefert.

Mit dieser Variante der Umsetzung ist der Server in der Lage, die *SPS* für die Fehlerverdeckung aller angeschlossener Clients zu verwenden, solange die Paketverlustraten sich im vorgesehenen Bereich von etwa 2% bewegen. Wird diese empirisch ermittelte Grenze überschritten und droht der Server aufgrund der höheren Berechnungszeit die isochrone Bearbeitungszeit zu überschreiten, erfolgt ein automatischer Rückfall auf die wenig komplexe *Noise Substitution*.

6.3.4.2 Erweitertes Verfahren beim Client

Die Verwendung der Basisverfahren beim Server verdecken Paketverluste auf dem Weg von den Clients zum Server, um das Mischsignal ohne Diskontinuitäten erzeugen zu können. Prinzipiell bietet es sich ebenso für die Verdeckung verlorener Pakete auf dem Rückweg vom Server zum Client an. Allerdings finden sich beim Client begünstigende Umstände wieder, die eine signifikant effektivere Bearbeitung ermöglichen. Die relevantesten Unterschiede zum Server sind dabei

Größere Verarbeitungszeit

Da der Client nur einen Audiodatenstrom zu bearbeiten hat, kann hier im Vergleich zum Server eine weit komplexere Bearbeitung erfolgen.

Kenntnis des eigenen Signals

Gegenüber dem Server verfügt der Client über partielle Information über die Audiodaten innerhalb eines Paketverlustes, nämlich über seine eigenen Daten, die er kurz zuvor zum Server gesendet hat. Diese Zusatzinformation lässt sich effektiv einsetzen, um die negativen Auswirkungen von Datenlücken weiter abzumildern.

Der erste Umstand erlaubt es uns, die Wahl des verwendeten Korrekturverfahrens hinsichtlich algorithmischer Komplexität beim Client weniger stark einzuschränken als beim Server. Bei der *SPS* ist dabei die Umsetzung mit einem vollständigen Suchparameterraum möglich, der zur Auffindung besser geeigneter Abschnitte führen kann.

Für die praktische Anwendung signifikanter ist der zweite Faktor, der im Gegensatz zum erst genannten nicht nur ein theoretisches Verbesserungspotenzial für bestimmte Situationen birgt, sondern eine kontinuierliche und effektive Erhöhung der Audioqualität bei Paketverlusten bietet.

Grundlage für die effektive Umsetzung dieser zusätzlichen Informationen ist die Verfügbarkeit des gemischten Audiosignals an einem Client A ohne den eigenen Beitrag. Dieser Restanteil $mix_{\tilde{A}}$ kann bei Kenntnis des eigenen Anteils am Audiomix s_A gewonnen werden, mit dem sein Audiodatenstrom am Server zum gemischten Signal beiträgt. Für jedes vom Server empfangene Sample $mix[t]$ kann Client A das korrespondierende Sample ohne seinen Anteil $mix_{\tilde{A}}[t]$ unter Abzug seiner Abtastwerte $samp_A[t]$ berechnen zu

$$mix_{\tilde{A}}[t] = mix[t] - samp_A[t] \cdot s_A$$

Da in NMP eine dynamische Positionierung eines Musikers auf der virtuellen Bühne (siehe Abschnitt 6.2) verwendet wird, sind jedem Client die beim Server verwendeten Mischparameter implizit bekannt und damit für die Rekonstruktion des partiellen Mischaudiodatenstroms $mix_{\tilde{A}}$ verfügbar. Der Aufwand für die Berechnung ist gering: neben dem lokalen (temporären) Vorhalten der eigenen Audiodaten $samp_A[t]$ im Speicher fällt nur eine geringe Rechenlast für die lineare Umwandlung an.

Zur Fehlerverdeckung muss beim Client die *SPS* dann nur auf $mix_{\tilde{A}}$ angewendet werden. Der so ermittelte Abschnitt $mix'_{\tilde{A}}$ wird mit den skalierten eigenen Audiodaten $samp_A[t] \cdot s_A$ überlagert und als Ersatz für die Datenlücke ausgegeben. Dieser Vorgehensweise folgend bezeichnen wir das beim Client durchgeführte Verfahren als *Partial Scaled Pattern Substitution* bzw. abgekürzt *PSPS*.

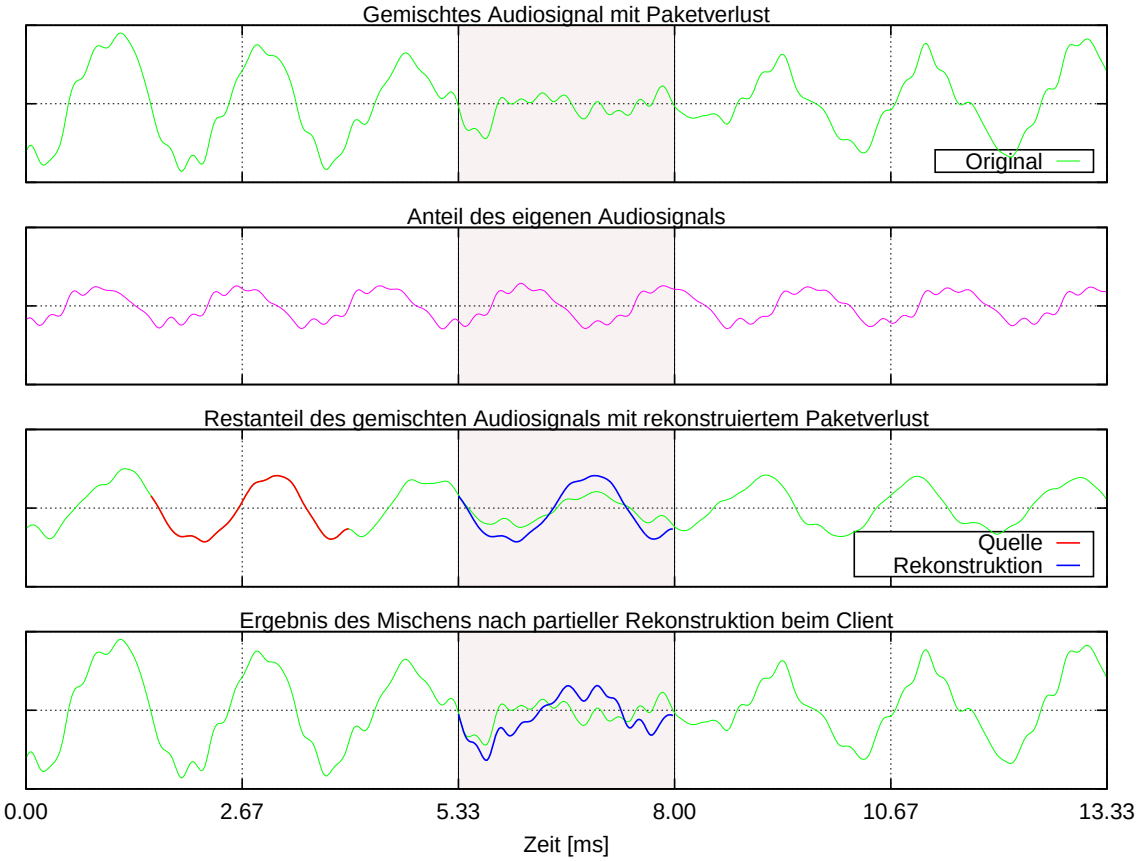

Abbildung 6.5: Partial Scaled Pattern Substitution beim Client

In Abbildung 6.5 ist das Ergebnis der Fehlerverdeckung beim Client durch die Anwendung des erweiterten Verfahrens dargestellt. Im obersten Diagramm ist das vom Server empfangene Mischsignal aufgetragen, mit einem angenommenen Verlust des hervorgehobenen Paketes. Der korrespondierende skalierte Eigenanteil $samp_A[t] \cdot s_A$ ist im folgenden Diagramm dargestellt. Mit dem Restsignal aller übrigen Teilnehmer $mix_{\tilde{A}}$ wird die *SPS* durchgeführt und ein geeigneter Ersatz für die Datenlücke ermittelt. Das abschließend beim Client gemischte und ausgegebene Audiosignal ist im unteren Diagramm aufgetragen, neben dem originalen Verlauf zum direkten Vergleich. Daraus ist die Abweichung des rekonstruierten Signals deutlich erkennbar, gleichzeitig sehen wir, dass eine Kontinuität gewahrt und der lokale Charakter erhalten bleibt. Auch ist sichtbar, dass der Client seinen Anteil im Mix deutlich wiederfindet und dadurch seinen Beitrag zum gehörten Audiosignal jederzeit und eindeutig identifizieren kann.

Somit verbessert die *PSPS* das subjektive Qualitätsempfinden bei gleich bleibenden Paketfehlerraten signifikant, da für jeden Client auch in Datenlücken das eigene Audiosignal unverfälscht erhalten bleibt. Dieser dient als konstanter Anker, der die lokale Charakteristik der Musik in Teilen konserviert und den Fokus der auditiven Wahrnehmung auf sich zieht. Zwar werden die verdeckten Passagen von den Benutzern nach wie vor registriert, das prinzipielle Manko des Basisverfahrens bei Mehrfachfehlern, das unter ungünstigen Bedingungen mitunter die Verfremdung der Kontinuität musikalischer Parameter bedingt, wird hierbei dagegen deutlich abgemildert.

Die Toleranzschranke erhöht sich bei Verwendung des erweiterten Ansatzes deutlich und ermöglicht das erfolgreiche Musizieren in Szenarien mit Paketfehlerraten von etwa 3%.

Darüber hinaus bietet es noch viel Raum für Optimierungen hinsichtlich einzelner Betriebsparameter. So wurden an unserem Institut exemplarisch Mechanismen wie das Verstärken des eigenen Audiosignals s_A bei gleichzeitiger Dämpfung des rekonstruierten Mischsignals, die Zuhilfenahme von Fading-Filtern oder die partielle Überlagerung von Rauschen an den Übergängen untersucht. Alle haben das Potenzial, subjektiv und situationsabhängig die wahrgenommene Qualität weiter zu verbessern, sind jedoch stark personenbezogen und damit nicht generalisiert anwendbar. Solche graduellen Verbesserungen stellen zukünftige Verfeinerungen dar und liegen außerhalb des Fokus dieser Arbeit.

6.3.5 Bewertung und Fazit

Wie alle interaktiven Internet Audioanwendungen, deren Reaktionszeit keine Neuübertragung verlorener Pakete erlaubt, muss NMP Paketverluste tolerieren und Lücken im Datenstrom für den Anwender möglichst unhörbar verdecken können.

Als typischer Vertreter interaktiver Audioanwendungen wird in der Literatur meist die IP-Telefonie behandelt, die jedoch mit signifikant abweichenden Betriebsparametern deutlich geringere Anforderungen an potentielle Fehlerkorrekturverfahren stellt. Als relevante Werte gelten hierbei die tolerierte Ende-zu-Ende Verzögerung im Bereich von 300 ms gegenüber 30 ms bzw. Blockgrößen zwischen 10 und 40 ms gegenüber 2.66 ms bei NMP.

Die geringe Latenztoleranz verhindert jede Anwendung von Verfahren, die eine zusätzliche Verzögerung erzeugen, und verbietet damit den Einsatz aller Mechanismen zur Vorwärtsfehlerkorrektur (*FEC*), mit denen eine Rekonstruktion verlorener Pakete möglich wäre, da diese immer mit einer zusätzlichen algorithmischen Verzögerung einhergeht.

Die im Vergleich zur IP-Telefonie bei NMP deutlich kleineren Blockgrößen erschweren außerdem die Anwendbarkeit etablierter Verfahren zur Fehlerverdeckung und die direkte Übertragung der Erkenntnisse daraus. Wir konnten lediglich die Ergebnisse für stationäre Ansätze bestätigen, indem wir die *Noise Substitution* als anwendbares Basisverfahren für die Fehlerverdeckung identifiziert haben, welches der *Silence Substitution* deutlich überlegen ist.

Alle darüber hinaus gehenden Verfahren, die auf einer Analyse des Signalverlaufs basieren, können für NMP nicht verwendet werden, da die ihnen zugrunde liegenden Algorithmen auf für IP-Telefonie typische Blockgrößen ausgelegt sind und für die in NMP verwendeten keine relevanten Ergebnisse liefern.

Aus der Erweiterung von Grundideen verschiedener bestehender Ansätze haben wir unser Verfahren für die *Scaled Pattern Substitution* entworfen, welches im Wesentlichen darauf abzielt, Datenlücken so zu ersetzen, dass die Kontinuität des Audiosignals erhalten bleibt. Dazu wird aus der unmittelbaren Vergangenheit des Signalverlaufs ein Abschnitt bestimmt, der bei Ersatz der Datenlücke eine minimale Abweichung der Amplitude und des Gradienten aufweist. Im Vergleich zur *Noise Substitution* verdoppelt die Anwendung dieses Verfahrens die Toleranz für anwendbare Paketfehlerraten auf etwa 2%. Das allen reaktiven Verfahren innewohnende prinzipielle Problem der Verfälschung momentaner Signalcharakteristik, wenn die Paketverluste an inhomogenen Stellen des Audiosignals auftreten, bleibt auch bei diesem Ansatz bestehen.

Dem begegnen wir für den Einsatz am Client mit einer Erweiterung dieses generell anwendbaren Verfahren zur *Partial Scaled Pattern Substitution*. Es nutzt das Vorhandensein der eigenen, zuvor dem Server übermittelten, Audiodaten aus, um den eigenen Anteil am ausgegebenen Audiosignal fehlerfrei zu rekonstruieren. Dazu errechnet der Client den Restanteil ohne seinen Beitrag am Mix, führt die *Scaled Pattern Substitution* auf diesen aus und mischt abschließend seinen eigenen Anteil hinzu. Die negativen Auswirkungen fehlende Pakete werden damit auf das externe Audiosignal beschränkt, während jeder Teilnehmer seine eigenen Audiodaten immer unverfälscht hört. Diese durchgängige Identifikation hat sich in unseren Untersuchungen

als so bedeutend erwiesen, dass die Musiker Paketfehlerraten von über 3% tolerieren und die Akzeptanz des NMP-Systems sehr hoch einstufen.

Neben der effizienten Fehlerverdeckung im Live-Modus muss unser System für die nachgelagerte Bearbeitung und Bereitstellung der aufgezeichneten Daten eine vollständige Korrektur aller zur Laufzeit fehlenden Audiopakete erlauben. Dafür haben wir einen Abschnitts weise agierenden Ansatz gewählt, der in Abhängigkeit von den verfügbaren Ressourcen die dem Server fehlenden Audiopakete entweder parallel zur Laufzeit erneut überträgt oder diese lokal archiviert und nach Beendigung der Sitzung im Ganzen übermittelt. Dieser Abgleich ist zeitunkritisch und wird daher in dieser Arbeit nicht detailliert behandelt.

6.4 Audiodatenkompression

Für das Musizieren in einem NMP-System wird qualitativ hochwertiges Audio benötigt, das in Form digitaler Audiodaten zu hohen Datenaufkommen führt. Wir haben in Abschnitt 5.6.1 festgestellt, dass für die Latenzminimierung eine möglichst hohe Abtastrate zu wählen ist und uns auf den Standardwert von 48 kHz für NMP festgelegt. Ein mit dieser Abtastrate digitalisierter Audiodatenstrom verursacht bei 16 bpS (*Bits per Sample*) Genauigkeit in Stereo eine konstante Datenrate von 1536 kbps. Der Server muss viele Clients gleichzeitig bedienen und bei Bedarf für die Simulation virtueller Bühnen individuellen Mehrkanal Surround-Sound für jeden Teilnehmer erzeugen und versenden. Ohne Datenreduktion der Audiodatenströme ist der Einsatz von NMP daher nur eingeschränkt oder gar nicht möglich, da die anfallenden Lasten die Netz- und Speicherkapazitäten schnell überschreiten.

In den folgenden Abschnitten untersuchen wir zunächst etwas genauer, welche Anforderungen die zu verwendenden Audiocodecs erfüllen müssen, um uneingeschränkt im NMP eingesetzt werden zu können. Wir werfen einen Blick auf heute verwendete Verfahren und begründen, warum sich keines davon für NMP eignet. Abschließend stellen wir einige Verfahren vor, die unseren strikten Latenzanforderungen genügen und vergleichen die Leistungsfähigkeit unter Berücksichtigung der eingangs aufgestellten Anforderungen.

6.4.1 Anforderungen an Audiocodecs

Auch bei der Wahl der im NMP zu verwendenden Audiocodecs ist die zentrale Anforderung das Einhalten minimaler Latenzen. Sie müssen das definierte Standardaudioformat von 48 kHz Abtastrate bei 16 Bit Genauigkeit unterstützen und in der Lage sein, mit Datenblöcken von 128 Samples zu operieren. Nur so kann gewährleistet werden, dass die aus der Audiohardware ausgelesenen Audiodaten unmittelbar weiterbearbeitet werden können. Um darüber hinaus ein latenzfreies Pipelining zu ermöglichen, dürfen keine Abhängigkeiten zwischen den codierten Datenblöcken entstehen.

Für die Wahl der erforderlichen Kompressionsraten muss neben der sonst üblichen Abwägung zwischen Datenrate und Audioqualität die isochrone Datenübertragung im Netz berücksichtigt werden. Die blockbasierte Arbeitsweise führt bei den meisten Audiocodecs dazu, dass jeder codierte Block aus einem Header und den Nutzdaten besteht. Der Header ist abhängig vom verwendeten Verfahren und kann für die Decodierung benötigte Daten zur Rekonstruktion enthalten, bspw. Sequenznummern, Prüfsummen oder dynamische Parameter. Dieser Overhead ist bei Verwendung höherer Blockgrößen vernachlässigbar, macht sich jedoch bei kleineren Werten nachteilig bemerkbar. Beim NMP kann diese durch die verwendeten kleinen Blockgrößen eine signifikante Verschlechterung der Relation zwischen Nutz- und Headerdaten verursachen. Dadurch, dass jedes Audiopaket über das Netz übertragen wird, werden jedem zusätzlich die Protokoll-Header der verwendeten Netzschichten angehängt. Im Falle einer Verwendung von RTP über UDP beträgt der minimale Protokoll-Header 40 Bytes. Mit einem Protokoll-Header von s_{h_N} und einem Codecheader s_{h_C} lässt sich die

Netzdatenrate N_r eines Audiodatenstroms D_r, der mit der Kompressionsrate r_C codiert wird, in Abhängigkeit der Blocklänge t_φ berechnen als

$$N_r = D_r \cdot r_C + \frac{s_{h_C} + s_{h_N}}{t_\varphi} \tag{6.1}$$

Damit beträgt der Anteil r_{AP} der Nutzdaten s_p am gesamten Datenaufkommen

$$r_{AP} = \frac{s_p}{s_p + s_{h_C} + s_{h_N}} \tag{6.2}$$

In Abbildung 6.6 sind die Kurven dieser Funktion in Abhängigkeit von der Kompressionsrate aufgetragen für einen einheitlichen Protokoll-Header von $s_{h_N} = 40$ Bytes mit den Größen für den Codecheader s_{h_C} von null, vier, 16 und 64 Bytes. Wir nehmen dabei das Standard Audiodatenformat von 48 kHz/16 bpS an, das mono vom Client zum Server übertragen wird. Mit einer Blockgröße von 128 Samples haben die Nutzdaten eine Länge von 256 Bytes. In unkomprimierter Form übertragen fällt kein Codecheader an, es wären lediglich die Protokoll-Header zu übertragen, womit jedes Paket eine Gesamtgröße von 296 Bytes hätte. Mit einer Paketrate von 375 Paketen pro Sekunde ergibt sich in diesem Fall eine Netzdatenrate von 888 kbps. Der Anteil der Nutzdaten am gesamten Datenaufkommen entspricht dabei knapp 86%. Grafisch lässt sich das an der obersten Kurve, die nur den Protokoll-Header berücksichtigt, mit dem Schnittpunkt des x-Wertes für unkomprimierte 16 bpS ablesen.

Wir erkennen weiterhin, wie ein etwaiger Codecheader das Verhältnis von Nutzdaten zum Overhead deutlich erhöht. Erfordert ein Verfahren bspw. 64 Bytes für den Codecheader, verringert sich der Nutzdatenanteil auf knapp 71%, der entsprechende Codec müsste eine Datenreduktion um Faktor 0.75 erreichen, um mit der Datenrate von unkomprimiertem Audio gleich zu ziehen. Dieses Missverhältnis verstärkt sich mit höheren Kompressionsraten sichtbar, da den konstanten Headerdaten immer weniger Nutzdaten gegenüberstehen. Bei einer Datenreduktion auf 1 bpS liegen die Nutzdatenanteile lediglich zwischen 13 und 28%, es ist somit nur bedingt sinnvoll, mit sehr hohen Kompressionsraten zu arbeiten. Unter der subjektiven Annahme, dass es wenig effizient ist, mehr Header- als Nutzdaten zu übertragen, ergeben sich für die Datenströme verschiedene Grenzwerte, bis zu denen eine Datenreduktion sinnvoll ist. Aus den Schnittpunkten der Kurven mit der Ordinate für einen Nutzdatenanteil von 0.5 geht hervor, dass ein Netzdatenstrom, der ohne zusätzliche Codecheader auskommt, bei einer stärkeren Kompressionsrate als 2.5 bpS mehr Header- als Nutzdaten enthält. Ist darüber hinaus je Paket ein Codecheader von 16 oder 64 Bytes zu übertragen, betragen die Grenzen sogar 3.5 bzw. 6.5 bpS.

Damit ergeben sich für den einzusetzenden Codec hinsichtlich des Datenformats und der Datenreduktion die Forderungen, dass ein möglichst kleiner Block Header verwendet wird und die Kompressionsrate in den Bereich zwischen zwei und vier bpS liegen sollte. Um Übertragungs- und Latenzverhalten vorhersagbar zu halten, muss darüber hinaus eine konstante Datenrate eingehalten werden, und zwar bezogen auf jeden einzelnen Block. Damit keine nachteiligen Varianzen bei der Verzögerung entstehen, darf jedes Paket eine vorgegebene Größe nicht überschreiten.

Bei dieser angestrebten Kompressionsrate muss die Audioqualität möglichst hoch sein, als Richtwert gilt der von der Audio-CD gewohnte Wert, wobei eine transparente Codierung angestrebt wird.

Zuletzt muss die Komplexität gültiger Verfahren möglichst gering sein und Codierung und Decodierung mit relativ gleichem Aufwand verbunden sein. Die Komplexität bestimmt maßgeblich die Verarbeitungsdauer in Form von CPU-Zyklen bei der Datenumwandlung. Da jeder Audiodatenblock auf dem Weg vom Client zum Server und zurück je zweimal codiert und decodiert wird, können bei zu hoher Komplexität der Verfahren zusätzliche Verzögerungen entstehen. Weiterhin bedingt das zentrale Mischen der Audiodatenströme am NMP-Server, dass hier pro angebundenen Client ein Audiodatenstrom zu decodieren und codieren ist. Die Komplexität des Codecs muss so gering sein, dass die Verarbeitung vieler paralleler Datenströme in Echtzeit

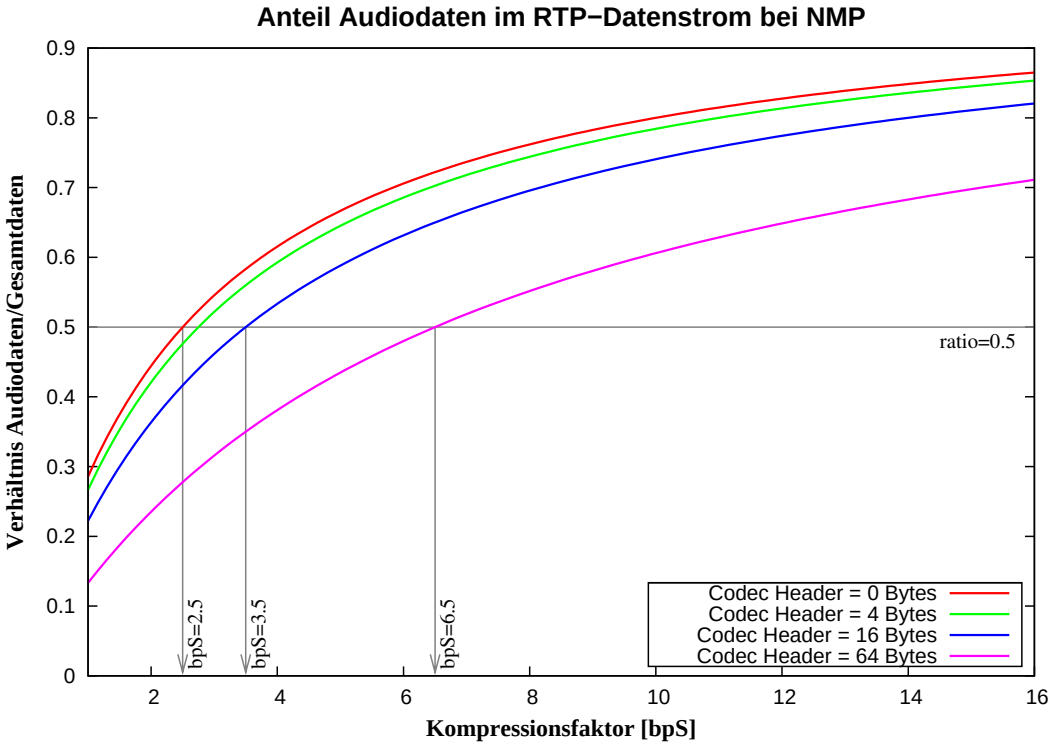

Abbildung 6.6: Anteil Audiodaten im RTP-Datenstrom beim NMP

möglich ist. Für die Abschätzung eines Richtwertes lehnen wir uns an die Kapazitätsgrenzen anderer Ressourcen an. So ist eine FastEthernet-Schnittstelle mit etwa 50 unkomprimierten Audiodatenströmen ausgelastet, das sichere und kontinuierliche Speichern auf Festplatte gelingt dauerhaft mit etwa 30 Audiokanälen. Diesen Wert gilt es nicht durch den verwendeten Codec zu unterschreiten, der somit in der Lage sein muss, mindestens 30 parallele Audiodatenströme in Echtzeit zu encodieren und zu decodieren.

Zusammenfassend müssen von den zu verwendenden Audiocodecs für den Einsatz im NMP folgende Kriterien erfüllt werden:

Audiodatenformat

Der von der Audiohardware eingelesene Datenstrom (48 kHz, 16 bpS) muss unterstützt werden. Dabei müssen mindestens Mono- und Stereokanäle verarbeitet werden können, für die Simulation virtueller Bühnen ist darüber hinaus Multikanalunterstützung wünschenswert.

Blockgröße und Intra-Codierung

Der Codec muss in der Lage sein, das Standardaudioformat in Blockgrößen von je 128 Samples zu verarbeiten. Dabei sollte die Länge des Codec Block Headers minimal sein, um einen hohen Nutzdatenanteil zu gewährleisten. Um das Pipelining nicht zu behindern, dürfen keine Abhängigkeiten zwischen den codierten Datenblöcken entstehen, jeder Block muss individuell codier- und decodierbar sein.

Kompressionsrate und Audioqualität

Um von Musikern akzeptiert zu werden, muss die Audioqualität beim NMP sehr hoch bis transparent sein. Als Maß für die Qualität ist ein ODG (*Objective Difference Grade*) nach PEAQ von mindestens -0.5 zu erreichen. Gleichzeitig ist bei der Verwendung des Standard Audiodatenformates eine Kompressionsrate von mindestens 4 bpS zu erreichen, eine weitere Reduktion auf mehr als 2 bpS verringert die Datenrate des RTP-Stroms nur unwesentlich. Höhere Kompressionsraten als 1:8 sind damit keine Be-

dingung.

Geringe Komplexität

Damit die Verarbeitung des Codecs beim Server nicht zum Engpass wird, darf der verwendete Codec nur eine geringe Komplexität haben. Anzustreben ist eine gleichzeitige Verarbeitung von etwa 30 Audiodatenströmen, die Codier- und Decodierzeiten sollten nicht zu weit voneinander abweichen und in Summe 30 mal schneller als Echtzeit ablaufen.

6.4.2 Aktueller Stand bei der Audiodatencodierung

Für die Datenreduktion im Audiobereich stehen prinzipiell die beiden Möglichkeiten Datenkonversion und Datenkompression zur Verfügung. Bei der Konversion werden die räumlichen oder zeitlichen Auflösungen der Audiodaten angepasst, wobei eine Datenreduktion dabei durch eine Verringerung der Abtastraten im ursprünglichen Datenstrom erreicht wird. Für NMP bietet es sich an, in der Audiohardware eine hohe Abtastrate zu wählen, um die Puffer schnell zu füllen, und anschließend in der Anwendung ein Downsampling (also eine Verringerung der Abtastrate) durchzuführen. Ein Downsampling um Faktor zwei halbiert dementsprechend das Datenaufkommen. Leider ist aus den in Abschnitt 5.6.1 gemachten Überlegungen der Spielraum für die Datenkonversion stark eingeschränkt. Für die räumliche Abtasttiefe von 16 Bit gibt es keine praktisch relevante Alternative, da acht Bit für Musik nicht ausreichend sind und Auflösungen zwischen diesen beiden Werten zu erheblichem Aufwand bei der Datenverarbeitung führen. Auch für die zeitliche Auflösung ist der Spielraum bei Musikanwendungen im wesentlichen auf die Werte 48 und 44.1 kHz beschränkt. Bereits eine Verringerung auf 32 kHz verschlechtert hörbar die Audioausgabe bei einer Reduktion der Datenrate um 33%. Als dritten Parameter kann die Anzahl der Kanäle angepasst werden. Ein Zusammenfassen beider Stereokanäle halbiert das Datenaufkommen, allerdings gehen dabei alle Positionsinformationen verloren. Die Konversion stellt damit ein praktisch anwendbares Werkzeug zur Datenreduktion dar, welches allerdings in unserem Fall eingeschränkt einsetzbar ist und zu erheblichen Qualitätsverlusten führt, da die Umwandlung den Inhalt des Audiosignals nicht berücksichtigt und lineare Operationen durchführt.

Darin unterscheidet sich die Datenkompression durch Audiocodierung. Hierbei werden die Audiodaten nicht Sample weise abgearbeitet sondern in größeren Einheiten von Blöcken oder Frames. Ein ganzer Block wird dabei semantisch analysiert und inhaltsabhängig platzsparend codiert. Die oben angesprochenen Konversionen sind teilweise bereits in den Audiocodecs integriert. So verwenden bspw. die meisten Verfahren die Mid/Side (bzw. Joint Stereo [52]) Codierung von Stereodaten, um Korrelationen zwischen den beiden Kanälen zu entfernen oder unterstützen Resampling. Es gibt heute eine Vielzahl von Verfahren für die Audiocodierung, die an unterschiedliche Umgebungen und Anforderungen angepasst sind und sich für den Einsatz im NMP entsprechend mehr oder weniger eignen.

Etablierte Audiocodecs lassen sich primär in zwei Gruppen unterteilen. Auf der einen Seite stehen die Verfahren, die für eine Bearbeitung von Musikdaten ausgelegt sind und einen Fokus auf hohe perzeptuelle Audioqualität legen. Das allgegenwärtige MP3 von Fraunhofer ist ein Vertreter dieser Musik Codecs, sein offizielle Nachfolger AAC codiert mittlerweile bereits bei einer Kompressionsrate von 0.1 transparent. Transparent bedeutet in diesem Fall, dass der ursprüngliche und der komprimierte Audiodatenstrom beim Abspielen für menschliche Zuhörer nicht unterscheidbar sind. Während sich diese Musikcodecs für latenzunkritische Anwendungen wie Archivierung oder Streaming bestens bewährt haben, werden bei interaktiven Anwendungen Sprach- bzw. Telefoniecodecs eingesetzt. Sie können sehr geringe Verzögerungen einhalten, sind jedoch meist auf die Bearbeitung von Sprache ausgelegt und bieten nicht die für eine Musikanwendung benötigte Audioqualität.

6.4.2.1 Audiocodierung von Musik

Die Audiocodecs, auf die man im Alltag bewusst trifft, sind für die Bearbeitung digitaler Musikdaten ausgelegt. Das von Fraunhofer entwickelte MP3 [16] ist mittlerweile ein Synonym für Online-Musik und allgegenwärtig beim Streamen von Audiodaten oder auf mobilen digitalen Musikplayern. Wie sein Nachfolger AAC und die meisten anderen hoch effizienten Verfahren arbeitet MP3 mit einem psychoakustischen Modell im Frequenzbereich. Nach der Umwandlung des digitalen Audiosignals aus dem Orts- in den Frequenzbereich können anhand des zugrunde gelegten Modells viele Datenanteile entfernt werden, ohne die auditive Perzeption zu beeinflussen. Kandidaten für die Entfernung dabei sind Frequenzen außerhalb des menschlichen Hörspektrums, nicht wahrnehmbare Obertöne oder Bereiche, die durch lautere Werte übertönt werden. Dem Herausfiltern irrelevanter Daten durch das psychoakustische Modell schließt sich die Entropiecodierung an.

Die Effizienz der Verfahren ist sehr hoch: unkomprimiertes Audio in CD-Qualität mit 44.1 kHz Abtastrate und 16 bpS in Stereo hat eine Datenrate von 1411 kbps. Die effizientesten heute verfügbaren MP3- und AAC-Encoder arbeiten bereits bei einer Zieldatenrate von 128 kbps transparent, d.h., auch bei einer Kompression um mehr als Faktor 11 können keine wahrnehmbaren Unterschiede zwischen Original und dem codiertem Audio herausgehört werden.

Verfahrensbedingt können Datentransformation zwischen Orts- und Frequenzbereich nur mit großen Datenblöcken durchgeführt werden. Beim MP3 beträgt daher die Blockgröße konstant 1152 Abtastwerte, mit der höchsten unterstützen Samplerate von 48 kHz beträgt die Pufferverzögerung 24 ms. Eine genauere Betrachtung der Verzögerungen unterschiedlicher Audiocodecs wird in [57] diskutiert und trifft auf alle gängigen Verfahren für die Audiocodierung von Musik zu. Damit scheiden diese für den Einsatz in NMP aus. Gleiches gilt für latenzoptimierte Anpassungen dieser Verfahren wie bspw. AAC-LD, der eine konstante und von der Abtastrate unabhängige Verzögerung von 20 ms bereitstellt. Damit ist der Einsatz von AAC-LD in interaktiven Anwendungen wie der Videotelefonie oder VoIP möglich, genügt den strikten Latenzanforderungen für NMP jedoch bei Weitem nicht.

6.4.2.2 Audiocodierung von Sprache

In Sprachanwendungen mit einem höheren Interaktionsgrad und geringerer Latenztoleranz (Telefonie, GSM, DECT, etc.) werden Sprachcodecs eingesetzt. Sie verzichten auf die zeitaufwendige Transformation in den Frequenzbereich und bearbeiten die Audiodaten im Zeitbereich. Dabei werden verschiedene Verfahren zur Vorhersage wie *linear predictive coding (LPC)* oder *Code-excited linear prediction (CELP)* verwendet. Diese Verfahren, die bspw. in [53] ausführlich erläutert werden, führen zu einer sehr hohen Effizienz, die in sehr guter Sprachqualität bei Kompressionsraten von bis hinunter zu 1:8 resultiert.

Als auf Sprachcodierung spezialisierte Verfahren verwenden die meisten dieser Codecs am Eingang 8 Bit logarithmischer PCM-Werte bei einer Abtastrate von 8 kHz, was zu der vom beim ISDN verwendeten G.711 bekannten Datenrate von 64 kbps führt. Die Blockgrößen und damit die Verzögerungen sind von Verfahren zu Verfahren verschieden und reichen von 0.25 ms beim G.726 (ADPCM) über 1.25 ms für G.728 (LD-CELP) bis hinauf zu 25 ms beim G.729 (CSA-CELP). Der Großteil der heute etablierten Verfahren verwendet Blockgrößen von mehr als 10 ms und ist daher für NMP ungeeignet.

Darüber hinaus eignen sich auch die Verfahren, die geringere Verzögerungen bieten, nur bedingt für unsere Anwendung. Zum einen ist das vorgesehene Audiodatenformat für Musik aufgrund der geringen zeitlichen und örtlichen Auflösung ungeeignet, zum anderen basieren die Vorhersageverfahren auf Annahmen, dass der zu bearbeitende Audiodatenstrom im menschlichen Stimmapparat entstanden ist. Bei der Bearbeitung von Musikdaten versagen die Modelle und führen zu unbrauchbarer Audioqualität.

6.4.3 PEAQ als Maß für Audioqualität

Als Maß für die Qualität wollen wir das von der ITU-T empfohlene und breit akzeptierte Verfahren PEAQ [46] (Perceptual Evaluation of Audio Quality) verwenden. Einen guter Überblick über PEAQ geben die Erfinder und Verfechter des Verfahrens in [92], zu denen gleichzeitig alle heutigen Anbieter kommerzieller Messanwendungen gehören. Das Verfahren stellt einen Kompromiss zwischen rein objektiven Verfahren wie dem PSNR [4] und subjektiven Bewertungen wie MOS [45] dar.

MOS steht für *Mean Opinion Score* und besagt bereits, dass eine Bewertung über eine Mittelung subjektiver Meinungen erfolgt. Die MOS-Skala reicht von *1* für *mangelhaft* bis hinauf zu *5* für *sehr gut*. Für die Gewinnung von MOS-Werten im Audiobereich sind aufwändige Hörtests in einer definierten Umgebung und einer großen Teilnehmerzahl erforderlich. Entsprechend existieren MOS-Werte gemeinhin für standardisierte und weitverbreitete Verfahren, für die ausreichend kommerzielles Interesse besteht, um den hohen Aufwand für die Tests zu rechtfertigen. Dazu fallen alle heute etablierten Audiocodecs, für die die jeweiligen Standardisierungsgremien wie ITU oder MPEG Hörtests als Teil der Standardisierung der Verfahren durchführen. Im Rahmen universitärer Forschung sind aussagekräftige Hörtests praktisch nicht durchführbar.

Ein objektives Maß wie der PSNR-Wert (*Peak Signal to Noise Ratio*) lässt sich dagegen leicht berechnen und gibt Auskunft darüber, wie stark sich der zu vergleichende Datenstrom vom Referenzdatenstrom unterscheidet. Er berechnet sich aus der mittleren quadratischen Abweichung, die anschließend auf den verwendeten Wertebereich normiert und logarithmiert wird. Zu einem Referenzaudiodatenstrom von n Abtastwerten $a_1 \dots a_n$ hat der Vergleichsdatenstrom $b_1 \dots b_n$ die mittlere quadratische Abweichung *MSE* (mean squared error) $MSE = \frac{1}{n}\sum_{i=1}^{n}(a_i - b_i)^2$. Für den PSNR wird diese anschließend mit dem Wertebereich ins Verhältnis gesetzt und logarithmiert, mit 16 Bit Abtastwerten gilt dann $PSNR = 20 \cdot log_{10}(65535) - 10 \cdot log_{10}(MSE)$.

Während PSNR nach wie vor ein wichtiges Maß bei der Beurteilung von Videocodecs ist, eignet es sich für die Audioqualität nur eingeschränkt. Besonders für die Bewertung effizienter Musik-Codecs, die anhand des verwendeten psychoakustischen Modells ganze Frequenzbänder bei der Codierung aus dem Audio entfernen, korrelieren die Werte von PSNR und subjektivem Eindruck nicht.

Diese Diskrepanz versucht das Verfahren PEAQ zu beheben. Anhand von aus Hörtests gewonnenen Erkenntnissen wurde ein Modell konstruiert, mit welchem eine objektive Beurteilung der Audioqualität vorgenommen wird. Das Modell dient dabei als Brücke zwischen dem objektiv gemessenen Abweichungen und der bei einem Zuhörer subjektiv zu erwartenden Beurteilung. Als Maß dient der ODG (Objective Difference Grade), der praktisch dem MOS mit Offset entspricht, so dass ein Wert von 0 einer transparenten Audioqualität entspricht, während am unteren Ende der Skala -4 eine starke Beeinträchtigung bedeutet. Das PEAQ Verfahren ist mittlerweile etabliert und wird von den meisten Audiocodecs bei der Bewertung der erreichten Qualität herangezogen. Insbesondere wird damit der Nachweis der Transparenz erbracht, für den die ODG Werte durchweg nahe Null liegen müssen und nie unter den Bereich von -0.5 fallen. Dadurch, dass die Grundannahmen für die Modelle beim PEAQ und in den psychoakustischen Audiocodecs ähnlich sind, erreichen Verfahren wie MP3 oder AAC schon mit relativ niedrigen Kompressionsraten transparente ODG Werte. Im Umkehrschluss bilden diese Verfahren somit die Messlatte, an die sich die übrigen Audiocodecs bei der Verwendung des PEAQ Verfahrens messen lassen.

Bei der Verwendung von PEAQ ist zu berücksichtigen, dass das Verfahren ausschließlich für hocheffiziente und hochqualitative Audiocodecs entwickelt wurde. Das äußert sich bereits daran, dass die Referenzdaten mindestens Audio-CD Qualität haben müssen, d.h., mindestens mit 16 bpS räumlicher Auflösung und 44.1 kHz zeitlich abgetastet sein müssen. Für die Beurteilung schmalbandiger Audiocodecs für die Sprachverarbeitung existiert mit PESQ [47] (Perceptual Evaluation of Speech Quality) ein alternatives Verfahren, das sich für unsere Anwendung nicht eignet.

Weiterhin werden bei PEAQ nur samplegenaue Codierungen berücksichtigt, die weder eine Änderung der Samplezahl noch eine Verschiebung der Samples innerhalb des Audiodatenstroms enthalten. Lücken im Datenstrom, wie sie bei NMP durch Paketverluste entstehen können, beeinflussen die Bewertung derart, dass die ODGs nicht mehr für eine zuverlässige Beurteilung der Audioqualität herangezogen werden können.

6.4.4 Echtzeitfähige Audiocodecs für NMP

Klammert man die betrachteten Sprach- und Musikcodecs für die Verwendung in NMP aus, engt das die Auswahl geeigneter Codecs stark ein. Neben der Möglichkeit, auf Audiodatenkompression vollständig zu verzichten, gibt es praktisch nur eine Hand voll Verfahren, die die strikten Latenzanforderungen für NMP erfüllen. Im Zuge der Entwicklung unseres NMP Prototypen haben wir dabei schrittweise die Verfahren ADPCM, FLAC und WavPack in unser System integriert und wollen an dieser Stelle einen kurzen Überblick über diese geben. Wir beschreiben zunächst die Merkmale und Besonderheiten jedes Verfahrens und schließen mit einer Evaluation der für den Einsatz in NMP relevanten Eigenschaften.

6.4.4.1 PCM

Solange wir uns im Labor bewegen und in lokalen Fast- bzw. Gigabit-Ethernet arbeiten, können wir auf eine Audiodatenkompression praktisch verzichten. Hinsichtlich Latenz ist die Verwendung des unkomprimierten Audiodatenstroms optimal, da die Daten unmittelbar verzögerungsfrei zwischen der Audio- und Netzschnittstelle ausgetauscht werden können.

Ein Audiodatenstrom vom NMP-Client zum Server hat in mono eine Datenrate von 768 kbps. Dazu kommen 120 kbps für den Protokoll-Overhead (40 Bytes·375 Pakete/s) und summieren sich zu 888 kbps je unkomprimierten Audiodatenstrom. Sendet der Server jedem Client einen individuellen und positionsgetreuen 3D-Audiodatenstrom, enthält das Mischsignal einen Audiokanal pro Teilnehmer n_C. Damit lässt sich die Netzlast abschätzen als $n_C \cdot (1 + n_C) \cdot 888\,kbps$, womit ein Fast-Ethernet Netz ausreichend ist, eine Sitzung mit bis zu neun Teilnehmern zu ermöglichen.

Gleichzeitig zeigt diese Abschätzung, wie zwingend nötig eine Datenreduktion für einen Einsatz von NMP in Netzen außerhalb des Labors ist.

6.4.4.2 ADPCM

Phoneme oder von Musikinstrumenten erzeugte Töne haben einen kontinuierlichen Signalverlauf, so dass benachbarte Abtastwerte innerhalb eines Audiodatenstroms üblicherweise stark korrelieren. Eine unabhängige Codierung jedes einzelnen Abtastwertes wie beim PCM Verfahren ist wenig effizient und kann dann, wenn man einen entsprechend kontinuierlichen Verlauf des Audiosignals erwarten kann, durch die Berücksichtigung der Ähnlichkeiten zwischen zeitlich benachbarten Samples zu einer effizienteren Datenkompression verwendet werden. Dieser Ansatz wird beim DPCM (Delta Pulse Code Modulation) verfolgt, bei dem nicht jeder Sample unabhängig, sondern die Differenz zum Vorhergehenden Wert codiert wird. Um die Kontinuität zu berücksichtigen, wird zunächst aufgrund der zurückliegenden Abtastwerte eine Vorhersage für den nächsten Abtastwert berechnet und nur die Abweichung des tatsächlichen Samplewertes von der Vorhersage als Restfehler codiert. Bei der Decodierung wird die gleiche Vorhersage gemacht und mit dem Restfehler der ursprüngliche Samplewert rekonstruiert.

Der Vorteil von DPCM liegt auf der Hand: solange benachbarte Samples einem monotonen Verlauf folgen, kann eine relativ genaue Vorhersage gemacht werden, so dass der Restfehler mit deutlich weniger Bits codiert werden kann als für die individuelle Codierung eines einzelnen Samples nötig sind. Die Anzahl der

Echtzeitfaktor		max. Leistung
Encodierung	Decodierung	[Datenströme/CPU]
621	734	336
Getestet mit 2GHz CPU		

Tabelle 6.1: Verarbeitungsgeschwindigkeit von ADPCM

zu verwendenden Bits für die Codierung des Restfehlers definiert im Wesentlichen die Kompressionsrate eines DPCM Verfahrens. Wird bspw. für einen Datenstrom mit 16 bpS eine Codierung des Restfehlers von 8 Bit gewählt, kann eine Datenreduktion auf Faktor 0.5 erreicht werden, bei 4 Bits auf entsprechen 0.25. Die verwendete Bitbreite ist für ein Verfahren konstant und führt bei ungünstigen Vorhersagen oder wenig monotonen Audiosignalen zu Überläufen, der Restfehler kann dann nicht mehr mit den verfügbaren Bits vollständig codiert werden. Diese Überläufe führen dazu, dass das Audiosignal nach der Decodierung deutliche Abweichungen hat und zu deutlich wahrnehmbaren Qualitätseinbußen führt.

Prinzipbedingt eignet sich DPCM eher für Sprachcodierung, weil hier ein relativ schmales Frequenzspektrum von 300 bis 3400 Hz abzudecken ist und von einer relativ starken Monotonie ausgegangen werden kann. Obertonreiche oder zischende Töne wie beim Schlag auf ein Becken, die von DPCM nur schlecht codiert werden, können mit dem menschlichen Sprechapparat nicht erzeugt werden. Das Verfahren hat daher in der Abwandlung als ADPCM (Adaptive DPCM), in denen die Quantisierungsstufen für die Codierung des Restfehlers dem Signal angepasst werden, Einzug in viele heute verwendete Sprachcodecs gehalten. Relevant hierbei ist der ITU-T Standard G.726 [43], der die ADPCM Codierung in der Sprachübertragung bei 40, 32, 24 und 16 kbps festlegt. Dieses Verfahren wird auch heute noch in Internetsprachanwendungen wie VoIP verwendet, da es sehr einfach implementiert und aufgrund geringer Komplexität auch auf einfachen Endgeräten eingesetzt werden kann.

Das ADPCM Verfahren funktioniert praktisch verzögerungsfrei, da für dir Vorhersage lediglich der vorhergehende Samplewert benötigt wird und damit Encoder und Decoder mit je einer Sampledauer (bei 48 kHz Abtastrate 20.8μs) auskommen.

Neben der Hauptanforderung einer geringen Verzögerung hat das Verfahren bereits aufgrund seines Alters eine sehr geringe Komplexität. Die entsprechende Messung wurde mit Hilfe eines repräsentativen Satzes von Audio-CD Titeln durchgeführt und dabei die in Tabelle 6.1 aufgeführten Werte ermittelt. Auf einem heutigen Standardrechner mit einer Rechenleistung von zwei GHz wurde bei der Codierung ein Geschwindigkeitsfaktor von 621 und bei der Decodierung von 736 ermittelt. Ein auf diesem Rechner laufender Prozess ist in der Lage, etwa 336 parallele Audiodatenströme in Echtzeit zu codieren und zu decodieren.

Mit der Latenzfreiheit und geringer Komplexität bietet sich somit dieses Verfahren für NMP an. Allerdings erfüllt die erzielbare Audioqualität bei ADPCM bei den angepeilten Kompressionsraten von 0.5 bis 0.25 nicht die eingangs aufgestellten Anforderungen für eine Musikanwendung. Typische MOS-Werte für den G.726 mit der Eingangsdatenrate von 64 kbps liegen bei etwa 3 für die Codierung mit 32 kbps und bei 2 mit 16 kbps.

Ungeachtet dessen ist das ADPCM Verfahren in NMP verwendbar in Szenarien, in denen eine entsprechende Verringerung der Netzdatenrate die Anwendung überhaupt erst praktikabel macht oder der Codec nur eine minimale Komplexität haben darf. Für solche Einsätze haben wir einen ADPCM Codec in NMP integriert, der das Standardformat 48k̇Hz / 16ḃit um Faktor vier von 768 auf 192 kbps netto komprimiert. Mit dem Overhead von 120 kbps beträgt dann die Netzlast 312 kbps und ebnet NMP den Weg zum Privatanwender in entsprechend dimensionierten Zugangsnetzen.

Echtzeitfaktor		max. Leistung
Encodierung	Decodierung	[Datenströme/CPU]
87	132	52
Getestet mit 2GHz CPU, Codieroption –fast		

Tabelle 6.2: Verarbeitungsgeschwindigkeit von FLAC

6.4.4.3 FLAC

Von der Xiph.org Foundation wird der freie verlustlose Audiocodec FLAC (Free Lossless Audio Codec, `http://flac.sf.net`) entwickelt und gehört mittlerweile aufgrund von nativer Unterstützung in Audioabspielgeräten und bei online vertriebener Musik zu den etablierten Verfahren. Bisweilen verdrängt es bei Angeboten, in denen Audio-CDs verlustlos über das Internet vertrieben werden, die kommerzielle Konkurrenz von Microsoft, Apple oder Real. Ein wichtiger Grund für seine Beliebtheit ist der vollständige Verzicht und kontinuierliche Prüfung auf patentierte Verfahren, so dass für die Verwendung keine Lizenzkosten anfallen. Der Vertrieb der Software unter der BSD-Lizenz gewährt privaten und kommerziellen Nutzern die kostenlose Verwendung des Verfahrens für beliebige Anwendungen zusammen mit der Gewissheit, nicht mit nachgelagerten Patentansprüchen konfrontiert zu werden.

FLAC unterstützt eine Vielzahl von Eingangsaudioformaten, darunter auch die für NMP relevante Codierung von 16 bpS bei 48 kHz Abtastrate und mit einer Blockgröße von 128 Samples. Damit können die Audiodaten beim Client unmittelbar nach dem Auslesen aus der Soundkarte codiert und über das Netz versendet werden, Pipelining ist ohne zusätzliche Pufferverzögerungen möglich. Zur Dekorrelation zwischen Kanälen ist in FLAC ein Mid/Side-Stereo Verfahren enthalten.

Die Codierung bei FLAC basiert auf zwei Vorhersagemodelle mit anschließender Entropiecodierung des Restfehlers. Zur Vorhersage werden entweder ein festes polynominales Modell verwendet, oder ein effizienteres, aus der Sprachcodierung bekanntes lineares Prädiktionsmodell (LPC). Die anschließende Entropiecodierung erfolgt mit Rice-Codes [77], wobei der vollständige Restfehler verlustlos codiert wird. Die Kompressionsrate lässt sich im Wesentlichen durch die Parametrisierung der Modelle derart beeinflussen, dass komplexere Vorhersagen eine bessere Näherung an das Signal ermöglichen und dadurch den zu codierenden Restfehler verringern. Gleichzeitig müssen die Modellparameter jeweils mit in den codierten Datenstrom abgelegt werden, so dass höhere Komplexitäten nicht zwangsläufig zu höheren Kompressionsraten führen. Eine genauere Betrachtung von FLAC ist in Salomons Werk [80] zu finden, das einen Überblick über alle etablierten Kompressionsverfahren gibt und die verwendeten Techniken umfassend erläutert.

Für unsere Anwendung ist die Verwendung von FLAC im einfachsten Modus am besten geeignet. In dieser Betriebsart mit der geringsten Komplexität wird die höchste Verarbeitungsgeschwindigkeit bei einer Kompressionsrate zwischen 0.5 und 0.65 erreicht. Mit eine komplexeren Vorhersagemodell lässt sich der Kompressionsfaktor lediglich um bis zu weitere 2% erhöhen. Bei einer Blockgröße von 128 Samples führt das zu einer Verkürzung des codierten Bitstroms um wenige Bits pro Block. Berücksichtigt man weiterhin, dass der Bitstrom immer auf volle Bytes aufgefüllt wird, ist das Potenzial komplexerer Modelle sehr begrenzt und rechtfertigt in keiner Weise die höheren Verarbeitungszeiten, die sich im Bereich von 50 bis 300% über denen, die die Codierung im einfachsten Modus benötigen.

Im schnellsten Codiermodus (Parameter `-fast`) haben wir für die im vorherigen Abschnitt beschriebene Komplexitätsmessung die in Tabelle 6.2 aufgeführten Werte ermittelt. Mit einer 87 bzw. 132 fach höheren Codier- bzw. Decodiergeschwindigkeit kann der NMP-Server etwa 52 Audiodatenströme in Echtzeit bearbeiten. Das ist zwar um fast eine Größenordnung langsamer als ADPCM, jedoch deutlich höher als der geforderte Wert von 30.

Prinzipbedingt ist die Kompressionsrate bei verlustloser Codierung abhängig von der Entropie des Audiosignals, die resultierende Bitrate des codierten Datenstroms ist daher variabel. Bei stark verrauschten Audiosignalen oder in lauten und disharmonischen Passagen (wie bspw. beim Beckenschlag) kann nur eine sehr geringe Datenreduktion erfolgen. Zusammen mit den Block-Header, der mindestens 14 Bytes beträgt, können so einzelne Audioblöcke die Größe der uncodierten Daten übersteigen. Diese Unvorhersagbarkeit der Bitrate verhindert den Einsatz von FLAC für NMP in Szenarien, in denen eine vorgegebene maximale Datenrate kontinuierlich nicht überschritten werden darf.

Damit eignet sich FLAC lediglich für Netze, die genügend Kapazität für den Transport des unkomprimierten Audiosignals haben, gleichzeitig aber mit der potentiellen Datenreduktion entlastet werden sollen. Eine Erweiterung des Einsatzradius von NMP auf schmalbandigere Netze ist mit FLAC nicht realisierbar.

6.4.4.4 WavPack

WavPack ist ein offener Audiocodec, der von David Bryant entworfen und entwickelt wurde. Ziel der Entwicklung war, ein vollständig freies und quelloffenes Verfahren zu schaffen, das vergleichbar leistungsfähig ist wie heute etablierte Verfahren. Das wird dadurch erreicht, dass ausschließlich lizenzfreie Verfahren und Algorithmen verwendet werden und gleichzeitig eine eingehende Prüfung und Ausschluss von patentierten Techniken betrieben wird. Die Quellen werden unter der BSD-Lizenz vertrieben und sind plattformübergreifend verfügbar für alle bekannten Betriebssysteme und eingebettete Systeme. Die Quelltexte und weitere Informationen können unter der Projekt-Seite `http://www.wavpack.com` abgerufen werden. Eine detaillierte Beschreibung des Codierverfahrens und der verwendeten Techniken ist ebenfalls in [80] enthalten, an dieser Stelle beschränken wir uns auf einen Überblick der für NMP relevanten Eigenschaften von WavPack und die Integration des Verfahrens in unser System.

WavPack unterstützt praktisch alle existierenden Eingabeformate für PCM codiertes Audio. Die Abtastraten können zwischen sechs und 192 kHz liegen, während die Abtastwerte als Ganzzahlwerte mit Genauigkeiten von 8, 16, 24 oder 32 bpS oder als Fließkommawerte verarbeitet werden. Es können beliebig viele Audiokanäle für die Codierung von Surround- und 3D-Sound in einem Datenstrom kombiniert werden, wobei eine Dekorrelation zwischen Kanälen im Mid/Side-Stereo Verfahren unterstützt wird. Die Blockgröße kann zwischen einem und 131072 Samples betragen. Damit erfüllt WavPack alle für NMP aufgestellten Anforderungen hinsichtlich Audioformat und Blockgröße.

Ähnlich wie FLAC ist auch dieser Codec primär für die verlustlose Kompression von Offline-Medien (bspw. Audio-CDs) vorgesehen. In diesem Modus ähneln sich die Leistungen beider Verfahren recht stark. Die Vergleichskriterien Kompressionsrate zu Codierzeit (resp. Komplexität) liegen nahe beieinander, Vor- und Nachteile zwischen Verarbeitungsgeschwindigkeit und Kompressionsrate sind abhängig von der Parametrisierung und wechseln sich von Einstellung zu Einstellung zwischen den beiden Verfahren ab. Im verlustlosen Modus hat auch bei WavPack der codierte Datenstrom eine variablen Bitrate und bedingt die gleiche Einschränkung des Verfahrens in Netzen mit begrenzter Kapazität. Hier spielt WavPack seinen verlustbehafteten und hybriden Modus als signifikanten Vorteil gegenüber FLAC aus.

Im verlustbehafteten Modus kann WavPack beim Encodieren eine Zieldatenrate vorgegeben werden, die es beim Bearbeiten nicht zu überschreiten gilt. Der Encoder beginnt mit der Integration der höchst signifikanten Bits in den Datenstrom und bricht beim Erreichen der vorgegebenen Grenze den Vorgang ab, die geringst signifikanten Bits werden nicht berücksichtigt. Dadurch wird gewährleistet, dass der codierte Datenstrom die vorgegebene Bitrate nicht überschreitet. Als positiven Nebeneffekt verringert der vorzeitige Abbruch der Verarbeitung die Ausführungszeit.

Die Zieldatenrate kann dabei bis hinunter zu 2.25 bpS gewählt werden, ein im NMP typisch verwende-

ter Datenstrom vom Client zum Server (mono, 16 bpS, 48 kHz) kann so von 768 kbps auf 108 kbps reduziert werden. Allerdings gilt das für den codierten Bitstrom, der bei den in NMP verwendeten Blockgrößen sehr stark durch die mitgeführten Block-Header verfälscht wird. Obwohl WavPack Blockgrößen bis hinunter zu einem einzelnen Sample unterstützt, ist die Codierung auf größere Blöcke ausgelegt und nur dann höchst effektiv. Denn unabhängig von der Blockgröße wird jedem codierten Audioblock ein Header von mindestens 64 Byte vorangestellt. Dieser dient zum einen zur richtigen Sequenzierung der Blöcke und der Fehlerprüfung des codierten Datenstroms, andererseits enthält er für den Decoder unverzichtbare Angaben über die Codierparameter und für die Rekonstruktion benötigte Vorhersagewerte.

Gemäß Formel (6.1) erhöht der Overhead aus Protokoll- und Headerdaten die Netzdatenrate auf 312 kbps. Der enormen Datenreduktion des codierten Bitstroms auf 108 kbps und der damit einhergehenden Reduktion der Audioqualität steht kaum mehr als eine Halbierung der unkomprimierten Datenrate gegenüber, eine Verwendung von WavPack in seiner nativen Form für NMP erscheint wenig sinnvoll. Dieses Problem haben wir mit einer Repaketisierung des WavPack-Headers wie in Abbildung 6.7 gelöst.

Den ersten Teil des nativen Headers bildet der Block-Header mit allen Informationen, die ein Parsen und Rekonstruieren von WavPack-codierten Datenströmen ermöglichen. Das beginnt mit der `ckID`, die ein vier Byte langen Delimiter in Form des menschenlesbaren Kürzels 'wvpk' enthält. In `ckSize` wird die Länge des gesamten codierten Blocks gespeichert, während `block_samples` die Anzahl der im Block enthaltenen Abtastwerte enthält. Das Feld `block_index` gibt die Sequenznummer des Blocks wider, aus dem sich die Sampleposition im Datenstrom bei konstanter Blockgröße berechnen lässt und die entsprechende Angabe in `total_samples` redundant wird. Einige Felder wie `track` und `index` sind der Bearbeitung ganzer Audio-CDs geschuldet, `version` und `flags` sichern die Interoperabilität verschiedener Versionen des Codecs ab. Zuletzt enthält `crc` eine Prüfsumme, um Bitfehler im codierten Datenstrom zu erkennen.

In unserer Anwendung werden nur die Angaben für die Sequenzierung und die Größe des jeweiligen Blocks benötigt. In den RTP-Header werden so lediglich die Werte von `block_index` und `ckSize` übernommen. Die Fehlerprüfung erfolgt innerhalb des Protokollstapels auf weiter unten liegende Schichten, so dass das Feld `crc` ignoriert wird. Da wir mit einer konstanten Blockgröße arbeiten, sind die Felder `cbSize`, `total_samples` und `block_samples` konstant bzw. können einfach aus dem `block_index` berechnet werden. Darüber hinaus sind die Felder `ckID`, `version`, `track`, `index` und `flags` konstant oder für unsere Anwendung irrelevant. Aus dem Block-Header sind somit nur zwei Felder zu berücksichtigen, die vollständig im RTP-Header untergebracht werden können.

Dem Block-Header schließen sich die Block-Metadaten an, die für die Decodierung benötigte Parameter und Vorhersagewerte enthalten. Während die Werte selber obligatorisch sind, enthält der Metadaten-Block für jeden Wert eine codierte Id und eine Längenangabe, die für ein vorgegebenes Audioformat konstant ist und berechnet werden kann. Von den vorhandenen Metadaten ist der Wert für `DECORR_TERMS` bei konstanter Blockgröße konstant. `DECORR_WEIGHTS` enthält für jeden Kanal einen 16 Bit-Wert, `DECORR_SAMPLES` und `HYBRID_PROFILE` je zwei 16 Bit-Werte und `ENTROPY_VARS` je drei 16 Bit-Werte pro Kanal. Die in `BLOCK_LENGTH` enthaltene Länge der Nutzdaten ist redundant, da sie sich aus der Angabe der Gesamtblocklänge `ckSize` im Block-Header berechnen lässt.

Nach dem Metadaten-Block schließt sich der codierte Bitstrom an.

Damit kann eine Repaketisierung wie in Abbildung 6.7 dargestellt wie folgt durchgeführt werden: auf der Senderseite wird der vom WavPack codierte Block ausgelesen und geparst. Aus dem Block-Header werden lediglich die Angaben über die Sequenznummer in `block_index` und die Blockgröße in `ckSize` in den RTP-Header übertragen. Aus den Metadaten werden die Werte für `DECORR_WEIGHTS`, `DECORR_SAMPLES`, `ENTROPY_VARS` und `HYBRID_PROFILE` entnommen und direkt hintereinander in den RTP-Payload abgelegt. Pro im codierten Datenstrom enthaltenen Kanal sind diese Werte in Summe 16 Byte lang. Der codierte Bit-

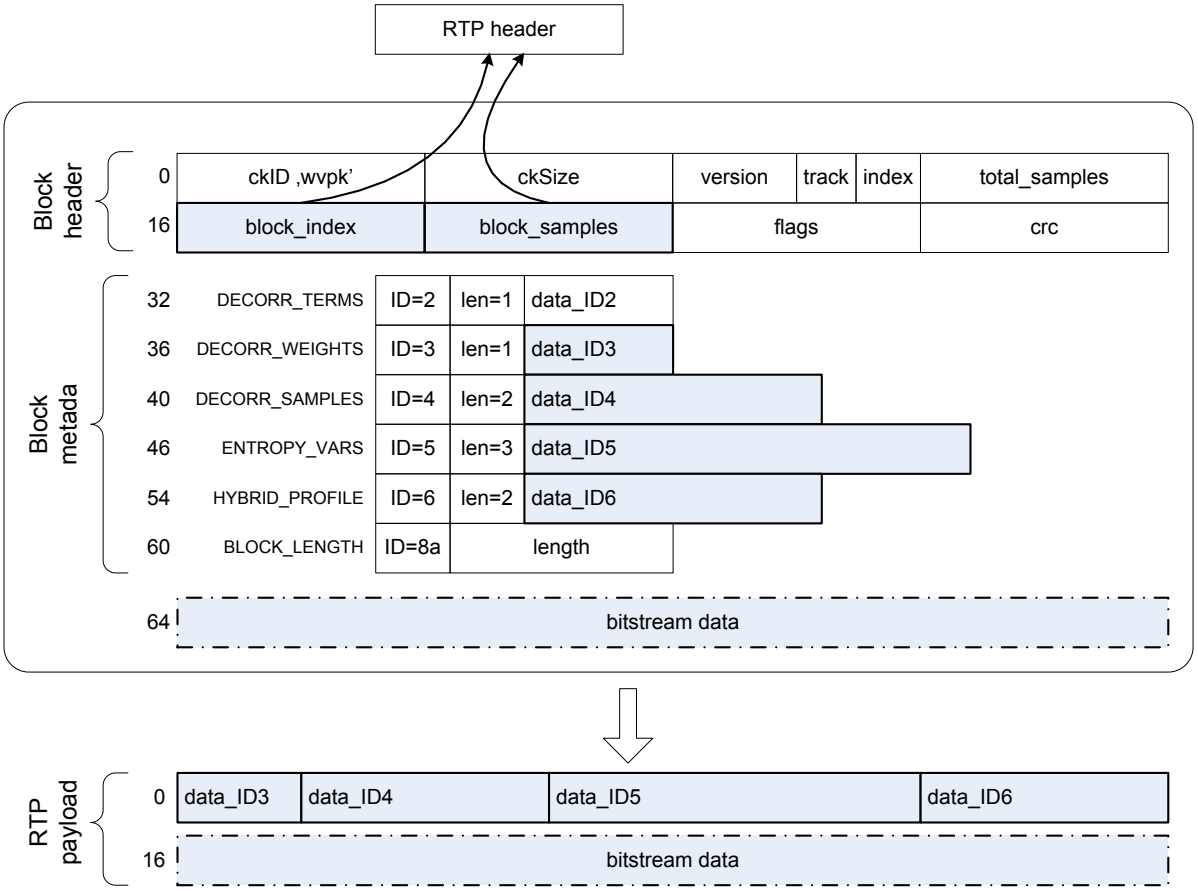

Abbildung 6.7: Repaketisierung des WavPack-Headers

strom wird anschließend direkt am Anschluss an die Metadaten in den RTP-Payload kopiert und das RTP-Paket versendet.

Am Empfänger wird diese Repaketisierung invertiert, um mit dem vorgegebenen Datenformat für den Codec konform zu bleiben. Anhand der Sequenznummer und der Länge der codierten Daten wird der ursprüngliche Blockheder rekonstruiert und die Werte der Metadaten an die richtige Position im Metadatenblock kopiert. Nach der Rekonstruktion des codierten Audioblocks wird dieser dem nativen WavPack-Decoder zugeführt.

Die beschriebene Repaketisierung verringert die Größe des Codecoverheads von ursprünglich 64 auf nun 16 Bytes pro Paket. Das führt zu einer enormen Steigerung der Effizienz, die sich besonders bei kleinen Blockgrößen wie in NMP und hoher Kompressionsrate stark bemerkbar macht. In einem typischen NMP-Setup bei einer Paketrate von 375 $\frac{Pakete}{s}$ beträgt der Overhead durch den Codec-Header statt der ursprünglichen 192 kbps nur noch 48 kbps. Zusammen mit dem Overhead von 120 kbps für Protokoll-Header (40 Bytes pro Paket) summiert sich die Datenrate des Overheads nun auf 168 kbps statt auf ursprünglich 312 kbps.

Mit diesem Verfahren haben wir den enormen Nachteil des hohen Header-Overheads von WavPack gegenüber FLAC beseitigt. Bei einer vorgegebenen Zieldatenrate von 2.25 bpS kann nun mit der Bitrate des codierten Datenstroms von 108 kbps und der Overhead-Datenrate von 168 kbps die Netzlast auf 276 kbps herabgesenkt und NMP zuverlässig auch in Netzen mit eingeschränkter Kapazität eingesetzt werden.

Darüber hinaus erweist sich die so genannte hybride Codierung von WavPack als ein für den Einsatz in NMP besonders gut geeignetes Verfahren für die On-Demand Dienste. Grundlage dieser erweiterten Dienste ist das Speichern und Archivieren aller während der Sitzungen produzierten und ausgetauschten Audiodaten.

Zieldatenrate [bpS]	Echtzeitfaktor Encodierung	Echtzeitfaktor Decodierung	max. Leistung [Datenströme/CPU]
2.25	102	154	61
3.00	93	137	55
4.00	82	120	48
6.00	69	95	39
8.00	61	81	34
verlustlos	59	78	33
hybrid 4.00	54	73	31
Getestet mit 2GHz CPU, WavPack 4.41 mit -fast Option			

Tabelle 6.3: Verarbeitungsgeschwindigkeit von WavPack

Im Gegensatz zum Echtzeitszenario ist hier die latenzunkritisch, da die Daten gestreamt werden. Wichtiger ist eine bestmögliche Audioqualität, idealerweise sollten die Audiodaten den Musikern in verlustloser Form zur Verfügung gestellt werden können. Hierfür bietet der hybride Modus von WavPack ein ideales Verfahren, um die Audiodaten in einen stark komprimierten Anteil für die Echtzeitsitzung und einem Korrekturanteil für die verlustlose Rekonstruktion des ursprünglichen Datenstroms aufzuteilen.

In diesem Modus codiert WavPack jeden Audioblock zunächst verlustbehaftet mit der vorgegebenen Zieldatenrate. Die restlichen weniger signifikanten Bits werden anschließend nicht verworfen, sondern in einem Korrekturdatenblock zurückgegeben. Der verlustbehaftet codierte Block kann anschließend zusammen mit dem Korrekturdatenblock vom Decoder verarbeitet werden und der ursprüngliche Audiodatenstrom verlustlos rekonstruiert werden.

Der Vorteil für NMP liegt dabei auf der Hand: statt in der Echtzeitsitzung mit einem latenzfreien Codec zu arbeiten und von jedem Client die Audiospuren anschließend mit einem Musikcodec in hoher Qualität codierte Daten zum Server zu senden, werden beim Musizieren die mit WavPack verlustbehaftet codierten Daten ausgetauscht und nach Sitzungsende lediglich die Korrekturdaten von allen Clients zum Server übertragen. Dort können die Audiospuren aller Teilnehmer vollständig rekonstruiert werden. Auch wenn wir im Rahmen dieser Arbeit die On-Demand Funktionalität innerhalb unseres NMP-Systems ausklammern, unterstreicht diese Eigenschaft von WavPack seine besondere Eignung für unsere Anwendung.

Bei der Messung der Komplexität spiegelt sich das Vorgehen bei der verlustbehafteten Codierung in den Zahlen wider. In Tabelle 6.3 sind die Vergleichswerte für WavPack aufgeführt, abweichend von den Messungen bei ADPCM und FLAC für unterschiedliche Zieldatenraten.

Dadurch, dass beim verlustbehafteten Codiervorgang der Ablauf mit steigendem Kompressionsfaktor immer früher abgebrochen und die restlichen Bits verworfen werden, arbeitet WavPack schneller, je niedriger die Zieldatenrate gewählt wird. Beim minimalen Wert von 2.25 bpS sind Codier- und Decodiergeschwindigkeiten 102 bzw. 154 mal höher als Echtzeit und ermöglichen so die parallele Verarbeitung von 61 Audiospuren. Bei der mit ADPCM erreichbaren Kompression von 4 bpS verringert sich dieser Wert auf 34. Ab einer Zieldatenrate von 8 bpS erhöht sich die Komplexität nicht wesentlich, zwischen den entsprechenden Faktor 34 und den für die verlustlose Codierung von 33 sind die Änderungen minimal. Der für den Einsatz in NMP relevante hybride Modus, bei dem der verlustbehaftete Teil eine Zieldatenrate von 4 bpS hat und die restlichen Bits in den Korrekturdatenstrom wandern, benötigt etwas länger, da Encoder und Decoder jeweils zwei Datenströme mit eigenständigem Header verarbeiten müssen. Mit Faktor 31 erfüllt die Verarbeitungsgeschwindigkeit von WavPack jedoch auch im hybriden Modus unsere Anforderungen.

6.4.5 Evaluationsergebnisse

Nachdem wir die Eigenschaften der vorgestellten Verfahren diskutiert und nachgewiesen haben, dass alle das essenzielle Kriterium Latenzfreiheit erfüllen und somit prinzipiell für NMP geeignet sind, stellt sich die Frage nach einen Vergleich bzw. einer Bewertung der Verfahren. Das bei Audiocodecs sonst üblicherweise verwendete Maß Effizienz als Quotient von erreichter Audioqualität und Kompressionsfaktor versagt hier, da wir es teilweise mit verlustloser Codierung zu tun haben. Wir können es nur für einen Vergleich zwischen ADPCM und WavPack im verlustbehafteten Modus heranziehen.

Die Ergebnisse eines solchen Vergleiches sind in Abbildung 6.8 abgebildet. Dargestellt sind die ODGs von PEAQ-Messungen von ADPCM und WavPack codierten Vergleichsdaten eines Audiostückes von 400 Sekunden Länge, das die meisten gebräuchlichen Musikrichtungen abdeckt. Gemessen wurde in Blöcken zu je einer Sekunde.

Zu erkennen ist zunächst einmal, dass die Qualität von ADPCM die gestellten Anforderungen deutlich unterschreitet. Der ODG-Wert unterschreitet mehrere Male den Wert von -3.0, was einer sehr schlecht bewerteten Audioqualität entspricht und damit die Verwendung von ADPCM in NMP auf Ausnahmesituationen limitiert. Bereits bei der höchsten Kompression von 2.25 bpS übertrifft WavPack die Qualität von ADPCM. In diesem Modus sind auch die Werte der stärksten Ausreißer oberhalb von -2.5, der mittlere Wert liegt im Bereich zwischen befriedigender und guter Qualität. Bereits mit einer Kompressionsrate von 3.0 bpS bleiben die Werte von WavPack vollständig oberhalb eines ODG von -1.0, was einer gute bis sehr gute Qualität entspricht. Schließlich erreicht das Verfahren mit 4.0 bpS Zieldatenrate, welche der des gegenübergestellten ADPCM entspricht, praktische transparente Qualität. Die Messwerte für diese Kompressionsrate belegen einerseits die Überlegenheit von WavPack gegenüber ADPCM und andererseits die auch für Musikanwendung völlig ausreichende Qualität von WavPack mit einem Kompressionsfaktor von vier und damit die sinnvolle Möglichkeit, NMP auch in Netzen mit geringer Kapazität mit sehr hoher Audioqualität zu betreiben.

Neben dem auf Qualität und Kompressionsrate fußenden Effizienzvergleich im verlustbehafteten Fall zwischen WavPack und ADPCM kann für die verlustlose Codierung FLAC praktisch nur über die Komplexität verglichen werden. Mit einer Verarbeitungsgeschwindigkeit vom 52 fachen der Echtzeit übertrifft es in dieser Disziplin WavPack mit dem entsprechenden Faktor von 33. Allerdings ist hierbei die Einschränkung zu berücksichtigen, wann eine verlustlose Codierung überhaupt einen Sinn macht. Die in diesem Modus erzielbaren durchschnittlichen Netto-Kompressionsraten zwischen 0.5 und 0.6 werden einerseits durch den Header-Overhead auf 0.6 bis 0.75 reduziert, andererseits erfordert die variable Bitrate einer solchen Codierung zum sicheren Betrieb mindestens eine Netzkapazität, die auch die unkomprimierten Audiodatenströme bewältigen kann. In solchen Fällen kann es sinnvoller sein, vollständig auf Audiodatencodierung zu verzichten und im Gegenzug von einer Ressourcenschonung durch geringere Komplexität zu profitieren und damit Rechenlast zu sparen.

Außerhalb des in dieser Arbeit behandelten Echtzeit-Szenarios bietet WavPack mit seinem hybriden Modus ein ideales Verfahren zur gleichzeitigen Abdeckung der On-Demand Funktionalität an und empfiehlt diesen damit als Standardverfahren für NMP.

6.4.6 Fazit

Neben der bandbreitenhungrigen Übertragung der Audiodaten in unkomprimierter Form als PCM, können nur wenige Audiocodecs die essenzielle Anforderung für NMP erfüllen und latenzfrei arbeiten. Etablierte Musikcodecs arbeiten mit psychoakustischen Modellen im Frequenzbereich und verwenden für ihre aufwändigen Analysen größere Datenblöcke, die die Verarbeitung um mindestens 20 ms verzögern. Telefoniecodecs

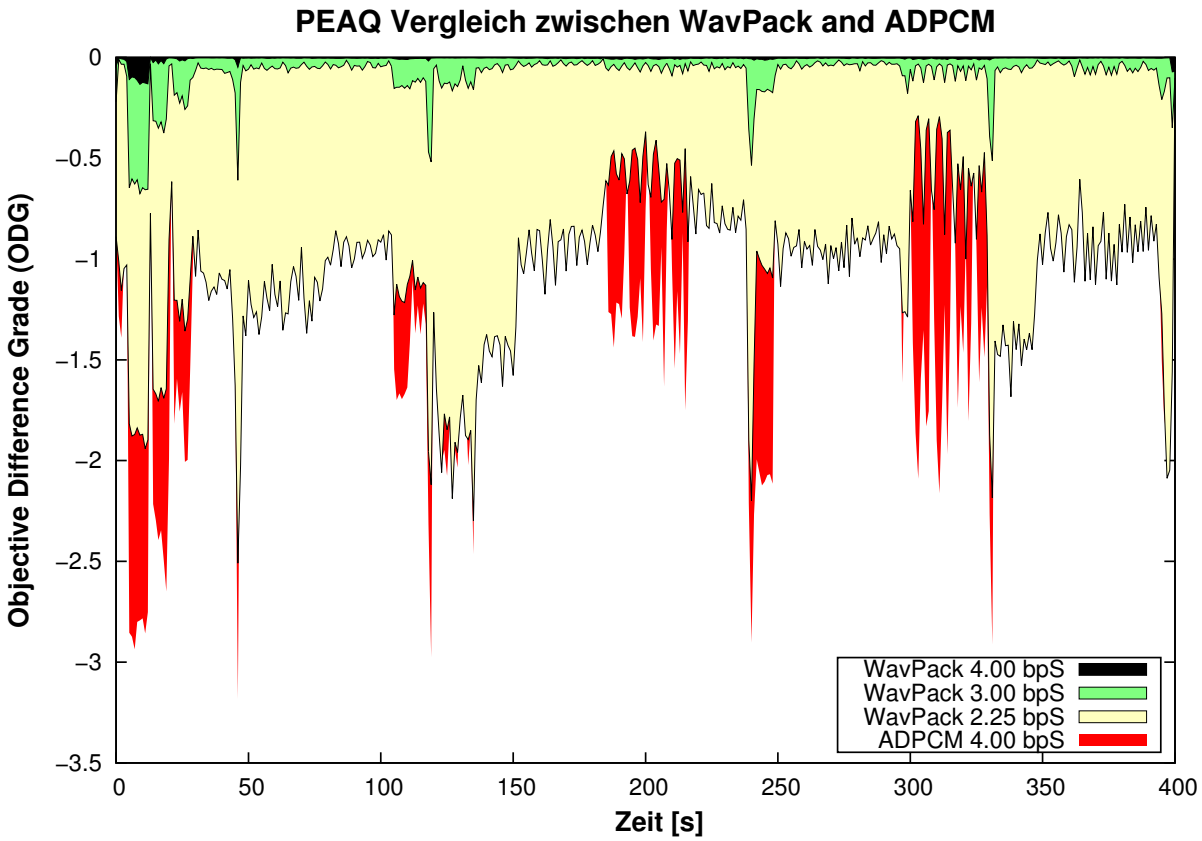

Abbildung 6.8: Vergleich der Audioqualität von WavPack und ADPCM

kommen mit deutlich geringeren Verzögerungen aus, sind jedoch auf das schmale Frequenzspektrum der Sprache zwischen 300 und 3400 Hz ausgelegt und eignen sich nicht für die Bearbeitung von Musik.

In diesem Abschnitt wurden das in der Sprachcodierung verwendete ADPCM Verfahren diskutiert und die abseits des Mainstreams verwendeten Codecs FLAC und WavPack vorgestellt. In Tabelle 6.4 sind die Eigenschaften dieser vier Verfahren gegenübergestellt.

Die Verarbeitung der Audiodaten als unkomprimiertes PCM stellt naheliegend das Optimum an Audioqualität und Komplexität dar, allerdings zum Preis von Datenraten, die NMP nur in breitbandigen Netzen einsetzbar machen. ADPCM als leichtgewichtiges und etabliertes Verfahren ist in der Lage, die Datenrate mit sehr wenig Rechenleistung auf ein Viertel zu senken, die Audioqualität ist für Musikanwendungen allerdings unbrauchbar. FLAC ist auf die verlustlose Kompression von hochwertigem CD Audio ausgelegt und kann in NMP für eine mittlere Datenreduktion um 30-40% verwendet werden. Allerdings ist die Bitrate variabel und ein sicherer Betrieb damit nur möglich, wenn die volle Netzkapazität für unkomprimiertes Audio verfügbar ist. Die Audioqualität bleibt dabei unberührt, für die Codierung und Decodierung eines Audiodatenstromes

Tabelle 6.4: Eigenschaften der in NMP eingesetzten Audio-Codecs

Codec	Kompressionsrate	Audioqualität	Komplexität
PCM	⊖⊖	⊕⊕	⊕⊕
ADPCM	⊕	⊖⊖	⊕
FLAC (verlustlos)	⊖/◯	⊕⊕	◯
WavPack (verlustlos)	⊖/◯	⊕⊕	⊖
WavPack (verlustbehaftet)	⊕	⊕	⊖/◯

verwendet eine aktuelle CPU etwa 2% ihrer Rechenleistung. Im verlustlosen Modus ähnelt WavPack sehr dem FLAC, ist jedoch noch etwas rechenintensiver und verwendet 3% der CPU. Im verlustbehafteten Modus bietet WavPack bei einer für jeden Audioblock garantierten Kompressionsrate von mindestens 1:4 eine nahezu transparente Audioqualität, bei einer dem FLAC Verfahren vergleichbaren Komplexität.

WavPack ist damit das geeignetste Verfahren, wenn es darum geht, NMP in schmalbandigen Netzen zuverlässig und mit sehr guter Audioqualität zu betreiben. Einen zusätzlichen Nutzen für die On-Demand Funktionalitäten in NMP beherbergt der hybride Modus. Obwohl dieser Modus die größten Vorteile auf sich und dem WavPack Verfahren vereinigt, bieten die drei Dimensionen Kompressionsrate, Audioqualität und Komplexität Spielraum, um in Abhängigkeit von Szenario und eingesetzter Hardware zwischen diesen Codecs zu wählen. Eine beispielhafte Richtlinie könnte wie folgt aussehen:

- Wenn die Netzlast keine Rolle spielt und das Netz eine hohe Kapazität hat
  ⇒ verwende PCM.
- Wenn das Netz über eine hohe Kapazität verfügt, die Netzlast aber trotzdem geschont und gleichzeitig mit verlustlosen Audiodaten gearbeitet werden soll
  ⇒ verwende FLAC
- Wenn die Netzkapazität beschränkt ist und die Audioqualität in den Echtzeit-Sitzungen gleichzeitig sehr hoch sein muss
  ⇒ verwende WavPack im verlustbehafteten Modus
- Wenn die Netzkapazität beim Client gering ist, der Server aber möglichst viele Anwender bedienen soll und die Qualität zweitrangig ist (Skype für Musiker)
  ⇒ verwende ADPCM

Solche Regelsätze lassen sich beliebig fein aufstellen und können dabei auch Differenzierungen für die Strecken von Client und Server und den Rückweg enthalten. Sie sind immer spezifisch für den Einsatz bzw. den Anforderungen der Anwender auszulegen. Für unsere Arbeit mit NMP hat sich die Verwendung von WavPack im verlustbehafteten bzw. hybriden Modus als beste Wahl erwiesen. Im folgenden wird, wenn nicht anders aufgeführt, dieses Verfahren verwendet. Die Freiheit, jederzeit die anderen diskutierten Codecs zu verwenden, bleibt davon unberührt.

6.5 Abspielverzögerung am Client

In Abschnitt 3.3 haben wir die Probleme von netzverteilten Echtzeit-Multimedia-Anwendungen diskutiert, die sich aus der variierenden Übertragungsverzögerung isochroner Daten ergeben.

Um beim Empfänger eine kontinuierliche Wiedergabe der Daten gewährleisten zu können, gibt es zur Kompensation der Schwankungen prinzipbedingt nur den Weg, die empfangenen Daten vor der Verarbeitung zunächst in einem Empfangspuffer zu sammeln. Erst wenn dieser Puffer einen vorgegebenen initialen Füllstand erreicht, wird mit der Bearbeitung der Daten begonnen.

Der initiale Füllstand stellt eine Reserve bereit, die Stauungen bei der Datenübertragung bis zu eben dieser Größe überbrücken kann. Erkauft wird die Reserve durch eine Erhöhung der Verweildauer der Daten im Empfangspuffer – die Kompensationsfähigkeit von Jitter kostet somit Latenz. Pakete, die mit noch höherer Verzögerung als die verfügbare Reserve empfangen werden, sind zum gegebenen Abspielzeitpunkt nicht verfügbar und werden verworfen. Die Dimensionierung des Empfangspuffers ist somit immer eine Abwägung zwischen Latenz und Paketverwurfswahrscheinlichkeit.

Entsprechend dem Einsatzzweck werden diese Puffer auch als *De-Jitter Puffer* bezeichnet. In Anwendungen, in denen der Sender die Rolle des Produzenten und der Empfänger die des Konsumenten von Multi-

mediadaten haben, werden die empfangenen Daten meist direkt nach dem Verweilen im Puffer ausgegeben. Diese Ausgabe in Form von Audio oder Video erfolgt über geeignete Abspielsoftware, entsprechend wird für den De-Jitter Puffer auch die Bezeichnung *Abspielpuffer* synonym verwendet.

Neben der Abwägung zwischen Latenz und Paketverlustrate spielen zahlreiche andere Faktoren bei der Dimensionierung des De-Jitter Puffers eine Rolle. Am naheliegendsten bestimmt diese der Grad der Interaktivität: handelt es sich um eine nicht-interaktive Anwendung ohne Anforderung an die Latenz, kann der Abspielpuffer beliebig groß gewählt werden. In der Praxis entspricht das dem Fall, wenn eine Mediendatei zunächst vollständig heruntergeladen und erst anschließend abgespielt wird. Dieser Extremwert für die Dimensionierung des Puffers macht aus der Streaming- eine Downloadanwendung.

Dieses Szenario deckt direkt einen anderen limitierenden Parameter auf: der Download der gesamten Datei erfordert entsprechende Ressourcen beim Empfänger – er muss in der Lage sein, die Daten vollständig auf Festplatte oder im Hauptspeicher abzulegen. In mobilen Geräten wird das selten der Fall sein, daher bestimmen dort die verfügbaren Ressourcen die Größe des Abspielpuffers in hohem Masse.

In interaktiven Anwendungen limitiert dagegen die maximal tolerierbare Systemlatenz die Größe des Abspielpuffers. Geben bspw. Erfahrungswerte für eine Telefonieanwendung vor, dass ein Gespräch nur mit einer Ende-zu-Ende Verzögerung von bis zu 300 ms sinnvoll geführt werden kann, dann ist diese Vorgabe ein hartes Limit, das bei der Dimensionierung des Abspielpuffers nicht überschreiten werden darf – unabhängig von der Paketverwurfswahrscheinlichkeit.

Eine derart harte Latenztoleranzgrenze in deutlich höherer Ausprägung haben wir auch bei NMP, so dass die maximale Ende-zu-Ende Verzögerung den gültigen Parameterraum stark limitiert. Für die Dimensionierung der Abspielpuffer in NMP haben wir einen statischen Ansatz und ein adaptives Verfahren entworfen, die in den folgenden Abschnitten erläutert werden.

6.5.1 Statischer Ansatz

Das statische Verfahren folgt der Forderung, während einer gesamten Sitzung eine gleichbleibende Systemlatenz innerhalb der vorgegebenen Toleranzgrenzen einzuhalten. Die gewählte Realisierung des Clients innerhalb der ISR der Audiohardware ermöglicht eine einfache und robuste Umsetzung dieses Ansatzes.

Die Ausführung des Clients wird periodisch durch den entsprechenden Interrupt getriggert, der das Auslesen des zuletzt aufgezeichneten und die Eingabe des nächsten auszugebenden Audioblocks erwartet. Die Realisierung einer gleichbleibenden Verzögerung zwischen ein- und ausgehenden Audiodaten ist mit den eingangs in Abschnitt 6.1 definierten Bedingungen für die Verarbeitung der Daten sehr einfach möglich: Der Client versieht jeden von der Audiohardware ausgelesenen Datenblock mit einer Sequenznummer, die laut Vorgabe über den gesamten Datenpfad bestehen bleibt. Sie wird ungeachtet aller Bearbeitungsschritte im Server und im Netz beibehalten, so dass der Client das Alter jedes empfangenen Paketes in Relation zum zuletzt von der Hardware ausgelesenen berechnen kann.

Für eine konstante Systemlatenz muss der Client daher jedes mal, wenn er von der Audiokarte den Block mit der Sequenznummer $readSeq$ ausliest das Audiopaket mit der Sequenznummer $writeSeq$ schreiben. Dabei bestimmt der Abstand $\Delta_{seq} = readSeq - writeSeq$ die Systemlatenz.

Mit dem Ergebnis aus der Latenzanalyse in Abschnitt 5.3.4, nach dem die Daten in der Audiohardware eine Verweildauer von $3t_\varphi$ haben, kann für eine gegebene maximale Latenztoleranz Δt_{max} und der bekannten Puffergranularität t_φ der maximal zulässige Wert für Δ_{seq} berechnet werden zu

$$\Delta_{seq} = \left\lfloor \frac{\Delta t_{max} - 3 \cdot t_\varphi}{t_\varphi} \right\rfloor = \left\lfloor \frac{\Delta t_{max}}{t_\varphi} \right\rfloor - 3$$

Für die bei NMP als Richtgröße angenommene maximale Latenztoleranz von 30 ms ergibt sich damit ein Abstand von $\Delta_{seq} = 8$. Äquivalent gilt dann durchgängig, dass bei jeder Aktivierung des Clients das Paket mit der Sequenznummer $readSeq$ gelesen und gleichzeitig das Paket mit der Sequenznummer $readSeq - 8$ zu schreiben ist, so wie es im Algorithmus 3 dargestellt ist. Wurde das Paket zu diesem Abspielzeitpunkt noch nicht empfangen, wird es verworfen. An seiner Stelle wird ein nach den in Abschnitt 6.3 beschriebenen Fehlerverdeckungsverfahren erzeugtes Paket eingefügt.

Algorithmus 3 : Statische Abspielverzögerung am Client

Input : Aufruf innerhalb der ISR
Input : Δ_{seq} konstant und durch maximale Latenztoleranz vorgegeben
lies Datenblock von Audiohardware;
packe Datenblock in Audiopaket($readSeq$) und sende es zum Server;
while *Audiopakete im Empfangspuffer der Netzschnittstelle vorhanden* **do**
  lies empfangenes Audiopaket ein;
end while
schreibe Audiopaket(($readSeq - \Delta_{seq}$)) in die Audiohardware;
$readSeq$++;

Dieser Algorithmus konzentriert sich einzig auf die Kernanforderung einer geringen und gleich bleibenden Latenz im NMP-System, die er auf eine einfache und deterministische Art sicher stellt. Ein Musiker, der regelmäßig NMP einsetzt, wird immer die gleiche Systemverzögerung vorfinden und sich daran gut gewöhnen können.

Darüber hinaus erfolgt bei diesem statischen Verfahren keine Abwägung anderer Betriebsparameter, wie bspw. der Audioqualität als zweitem wichtiger Kriterium für ein erfolgreiches gemeinsames Musizieren. So birgt dieser Ansatz, trotz seiner Einfachheit und Effizienz, mit der bedingungslosen Fixierung auf die Systemlatenz einige Schwächen in bestimmten Szenarien, von denen als naheliegendste die folgenden zu nennen sind:

1. Hat ein Client eine mittlere Ende-zu-Ende Verzögerung von $\Delta_{seq} \cdot t_{\varphi}$, führt jede Variation der Übertragungsverzögerung zu einem verspäteten Paket. Das resultiert in einer hohen Paketverlustrate und äquivalent zu einer schlechten Qualität. In solchen Szenarien kann eine leichte Anhebung der Latenz eine signifikante Erhöhung der Audioqualität herbeiführen.
2. Clients, deren Übertragungsverzögerungen über den festgelegten Wert für die Abspielverzögerung liegt, können an NMP Sitzungen gar nicht teilnehmen.
3. Ein in unmittelbarer Netzdistanz am Server angebundener Client mit durchweg geringer Ende-zu-Ende Verzögerung kann dagegen bei diesem Verfahren nicht von einer geringeren Latenz Gebrauch machen.

Offenkundig ist für die Behebung dieser Schwächen eine dynamische Dimensionierung der Abspielverzögerung erforderlich, die zwischen Systemlatenz und Paketverlustrate und der daraus resultierenden Audioqualität abwägt. Einen solchen adaptiven Ansatz in unterschiedlichen Ausprägungen stellen wir im folgenden Abschnitt vor.

6.5.2 Adaptive Abspielverzögerung

Für ein adaptives Verfahren wird ein Bewertungsmaß benötigt, um zwischen Ende-zu-Ende Verzögerung und Audioqualität abwägen zu können. Dieses Maß muss in der Lage sein, für eine gegebene Latenz die Paketverlustwahrscheinlichkeit zu bestimmen.

Die Abhängigkeit der beiden Stellparameter gewinnen wir aus statistischen Analysen, die wir kontinuierlich während der gesamten Laufzeit einer Sitzung vornehmen.

6.5.2.1 Herleitung des Bewertungsmaßes

Die Herleitung unseres Verfahrens beruht darauf, dass wir während einer Sitzung über kurz- und mittelfristige Zeitspannen gleich bleibende Netzeigenschaften annehmen und damit Abschätzungen aus unmittelbar zurückliegenden Messungen herleiten können. Diese Annahme ist durchaus begründet, da wir für eine praktische Durchführung von NMP Sitzungen solch hohe Ansprüche an die Zuverlässigkeit des Netzes stellen. Daher ist für hinreichend kleine Messzyklen die Extrapolation der zuletzt ermittelten Ergebnisse auf den nachfolgenden Beobachtungszeitraum zulässig.

In unserem adaptiven Verfahren wird die Verweildauer $t_{NS}(p_i)$ jedes Audiopaketes außerhalb des Clients kontinuierlich ermittelt und statistisch ausgewertet. Die Berechnung ist anhand der persistenten Sequenznummer i ähnlich direkt wie im statischen Verfahren möglich. Dabei werden bei jeder Aktivierung des Clients, neben dem Datenaustausch mit der Audiohardware, alle in den Empfangspuffern der Netzschnittstelle verfügbaren Audiopakete ausgelesen.

Dadurch, dass die Aktivierung der Verarbeitung zu zeitlich äquidistanten Abständen von t_φ erfolgt, können wir die Verweildauer t_{NS} eines Paketes p_i mit eben dieser Granularität messen, es gilt dann

$$t'_{NS}(p_i) = \left\lceil \frac{t_{NS}(p_i)}{t_\varphi} \right\rceil \cdot t_\varphi = \Delta_{seq}(p_i) \cdot t_\varphi$$

Statt mit Zeitmessungen zu arbeiten, reicht es die Differenz der Sequenznummern des zuletzt von der Soundkarte gelesenen Datenblocks $readSeq$ mit der Sequenznummer des von der Netzinterface gelesenen Paketes $seq(p_i) = i$ zu bilden. Diese Differenzen $\Delta_{seq}(p_i) = readSeq - seq(p_i) = readSeq - i$ werden wie in Algorithmus 4 für die Berechnung der Häufigkeitsverteilung ausgewertet.

Algorithmus 4 : Berechnung Histogramm über Paketlatenz

Parameter : Δ_{seq} = Paketlatenz
Parameter : seq = Sequenznummer des Paketes

Data : actH = aktuell berechnetes Histogramm
Data : lastH = Histogramm der letzten Messperiode
Data : n_H = Anzahl Messungen pro Periode

```
actH[Δ_seq]++
if ( seq mod n_H ) == 0 then                    /* Ende der Messperiode erreicht? */
    lastH = actH                          /* speichere aktuelles Histogramm */
    actH.clear()                                /* lösche Statistiken */
end if
```

Dabei wird in Abschnitten zu je n_H Messungen kontinuierlich ein Histogramm H' erstellt, welches die absoluten Häufigkeiten der Paketlatenzen Δ_{seq} in Klassen konstanter Breite t_φ zählt. Sobald ein Messabschnitt abgeschlossen ist, wir das errechnete Histogramm H' nach H kopiert, welches innerhalb der nächsten Periode als Entscheidungsgrundlage dient.

$H(b)$ enthält die Anzahl der Pakete während der letzten Messperiode, deren Verweildauer außerhalb des Clients $b \cdot t_\varphi$ betragen hat. Sie gibt unmittelbar Aufschluss über die Verteilung der Paketlatenzen und erlaubt einen direkten Rückschluss auf die Güte der Datenübertragung. Liegen die Werte alle innerhalb weniger benachbarter Klassen, handelt es sich um eine ausgeglichene und für NMP vorteilhafte Verbindung, während eine breite Streuung auf einen hohen Jitter hinweist.

Für unsere Bewertungsfunktion betrachten wir nicht die Einzelwerte, sondern interessieren uns für den Anteil der Pakete, die mit einer vorgegebenen Abspielverzögerung abgespielt werden können. Dieses Verhältnis gewinnen wir aus dem kumulierten Histogramm $H_c(k)$, das die Summe der Einzelwerte für alle Klassen

von Null bis k enthält, die anschließend durch die Anzahl der Messungen n_H dividiert wird. Der resultierende Wert

$$H_c(k) = \frac{\sum_{i=0}^{i=k} H(i)}{n_H}$$

gibt dann den relativen Anteil der Pakete in der zurückliegenden Messperiode wieder, die eine Verweildauer außerhalb des Clients von weniger als $k \cdot t_\varphi$ hatten.

Äquivalent gilt für die Paketverlustwahrscheinlichkeit L bei der Abspielverzögerung $k \cdot t_\varphi$ von $L(k) = 1 - H_c(k)$. Diese Berechnung versetzt uns jederzeit in die Lage, eine Vorhersage über die Audioqualität für eine beliebige Abspielverzögerung zu treffen.

6.5.2.2 Konstruktion einer Basis Präferenzmatrix

Mit dem Bewertungsmaß $L(k)$ wissen wir nun zwar, wie für den betreffenden Client in der laufenden Sitzung Latenz und Qualität voneinander abhängen. Eine Abwägung zwischen den beiden Parametern ist dagegen – wie so oft bei NMP – rein subjektiv. So kann für den Perkussionisten die Audioqualität einer geringen Latenz gegenüber unbedeutend sein, da für ihn die hohe Interaktion ausschlaggebend sind. Ein Pianist wird das eher genau anders herum sehen.

Mangels universell gültiger Formeln haben wir für die Abwägung einen flexiblen und personalisierbaren Ansatz gewählt. Die Grundlage bildet qualitativ die allgemeingültigen Annahmen ab, die quantitative Justierung erlaubt eine individuelle Anpassung der Abwägung.

Als allgemeingültig fassen wir dabei auf, dass die Zufriedenheit eines Musikers mit einem NMP-System am höchsten ist, wenn es ohne Qualitätsverluste und Latenz arbeitet. Die Zufriedenheit nimmt kontinuierlich mit steigender Paketverlustrate und Latenz ab. Folgende Annahmen haben wir, basierend auf Erfahrungen mit Musikern während unserer Arbeit, dabei getroffen:

1. Der für die Latenz angepeilte Wert von 30 ms stellt für die meisten Musiker eine gut tolerierbare Grenze dar. Kleinere Latenzen werden positiver bewertet, wobei die Verbesserungen mit abnehmender Latenz geringer eingestuft werden. Steigt die Latenz auf der anderen Seite über 30 ms, nimmt die Zufriedenheit und Bedienbarkeit von NMP deutlich ab. Im unspielbaren Bereich verliert zusätzliche Latenz an Bedeutung.
2. Ein ähnlicher Verlauf der Zufriedenheitskurve ergibt sich für die Audioqualität: die verwendeten Mechanismen zur Verschleierung von Paketverlusten sorgt dafür, dass bis zu Verlustraten von 1% die Zufriedenheit nur langsam abnimmt. Bei etwa 2% sind die Störungen bereits deutlich wahrnehmbar, darüber hinaus wird die Qualität zunehmen schlecht und ab etwa 4% als unannehmbar bewertet.

Somit ähneln sich die Verläufe der Zufriedenheitskurven für die beiden Stellparameter bei isolierter Betrachtung. Sie haben beide einen zum Mittelpunkt symmetrischen S-förmigen Verlauf, der einem allmählichen Übergang von 1 nach 0 folgt. Unter anderem erfüllen die Kreisfunktionen innerhalb vorgegebener Definitionsbereiche diesen Verlauf. So haben wir als Basis für die fallende Zufriedenheitskurve die Cosinusfunktion $\frac{(cos(x \cdot \pi)+1)}{2}$ im Bereich $[0 \ldots \pi]$ gewählt, die wir auf den Wertebereich normiert haben, womit sich der in Abbildung 6.9 dargestellte Verlauf ergibt.

Die resultierenden Kurven für Latenz f_L und Qualität f_Q werden analog dazu auf die ermittelten Wertebereiche ausgedehnt und anschließend zu einer Bewertungsmatrix kombiniert. Die Paketverlustwahrscheinlichkeit als vom System nicht beeinflussbarer Parameter wird als Abszisse verwendet, die Latenz als Ordinate. Die Abhängigkeit zwischen $f_Q(x)$ und $f_L(y)$ wird als Komposition des arithmetischen und geometrischen Mittels abgebildet und die symmetrische Zufriedenheitsmatrix $f_Z(x, y)$ wie folgt definiert:

$$f_Z(x, y) = \frac{\sqrt{f_Q(x) \cdot f_L(y)} + \sqrt{f_Q(x) + f_L(y)} \cdot 0.3}{1 + \sqrt{2} \cdot 0.3}$$

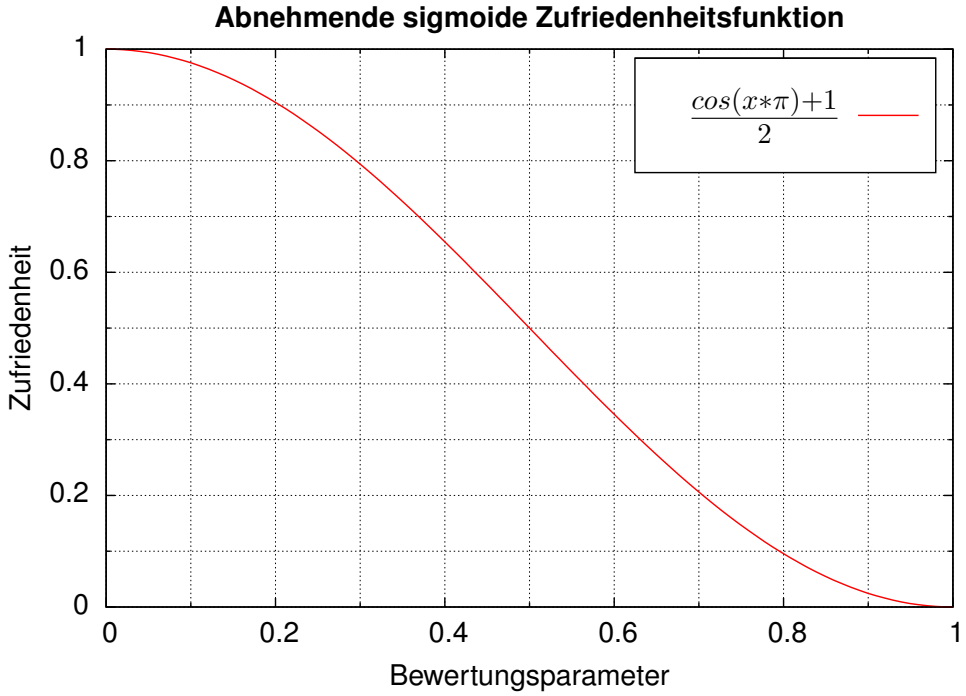

Abbildung 6.9: Normierte Zufriedenheitsfunktion für einen Stellparameter

Für eine sinnvolle Verwendung der Matrix als Entscheidungskriterium wird eine diskretisierte Form benötigt, die beide Dimensionen in Klassen einteilt. Für die Latenz ist diese durch die Granularität in Einheiten zu t_φ inhärent gegeben. Die Granularität der Paketverlustwahrscheinlichkeit ist ebenfalls begrenzt, da die Berechnung der relativen Häufigkeiten den Wertebereich in Einheiten zu n_H^{-1} aufteilt. Im Verlauf unserer Arbeit hat sich eine Auflösung der Paketverlustwahrscheinlichkeit von 2‰ für die Entscheidungsfindung als ausreichend erwiesen. Werte, die den vorgesehenen Bereich überschreiten (d.h., Latenzen von mehr als $20 \cdot t_\varphi$ und Paketverlustraten von mehr als 4%) werden auf die Maximalwerte limitiert.

Eine so aufgelöste symmetrische Bewertungsmatrix $f_Z(x,y)$ ist in Abbildung 6.10 dargestellt. Die x-Achse teilt die Paketverlustwahrscheinlichkeit zwischen Null und 4% in 20 gleiche Klassen auf, während die y-Achse die Latenz in Vielfache von t_φ zwischen eins und 20 abbildet. Die Zufriedenheit ist bei geringen Werten für Latenz und Störungen hoch und nimmt oberhalb der genannten Grenzen markant ab. Aufgrund der Wahl von Kreisfunktionen als Bewertungskurven ergibt sich ihre Kreissegment förmige Ausprägung.

Die auf der Bewertungsmatrix basierende adaptive Dimensionierung des Abspielpuffers verläuft in drei Schritten:

Berechnung der Latenz/Qualität Wertepaare

Basis für eine Bewertung ist die Berechnung von Wertepaaren aus Latenzen $m_L(i)$ und korrespondierenden Paketverlustraten $m_Q(i)$. Aus obiger Betrachtung wissen wir, dass wir mit dem verfügbaren Histogramm $H(i)$ die benötigte Liste einfach berechnen können zu $M_Z = \{(0,L(0)),(1,L(1)),\ldots,(n,L(n))\}$. Ein Wertepaar $M_Z(k) = (k,L(k))$ gibt die in der zurückliegenden Messperiode realisierbare Paketverlustrate bei der Verwendung eines Abspielpuffers von k Paketen wieder. Diese Liste berechnen wir für alle relevanten k mit $1 \leq k \leq 20$.

Bewertung der Wertepaare

Für jedes zu berücksichtigende Wertepaare wird in der Bewertungsmatrix der Zufriedenheitswert $m_Z(i) = f_Z(m_L(i), m_Q(i))$ nachgeschlagen. Dieser gibt im Nachhinein die Bewertung für alle poten-

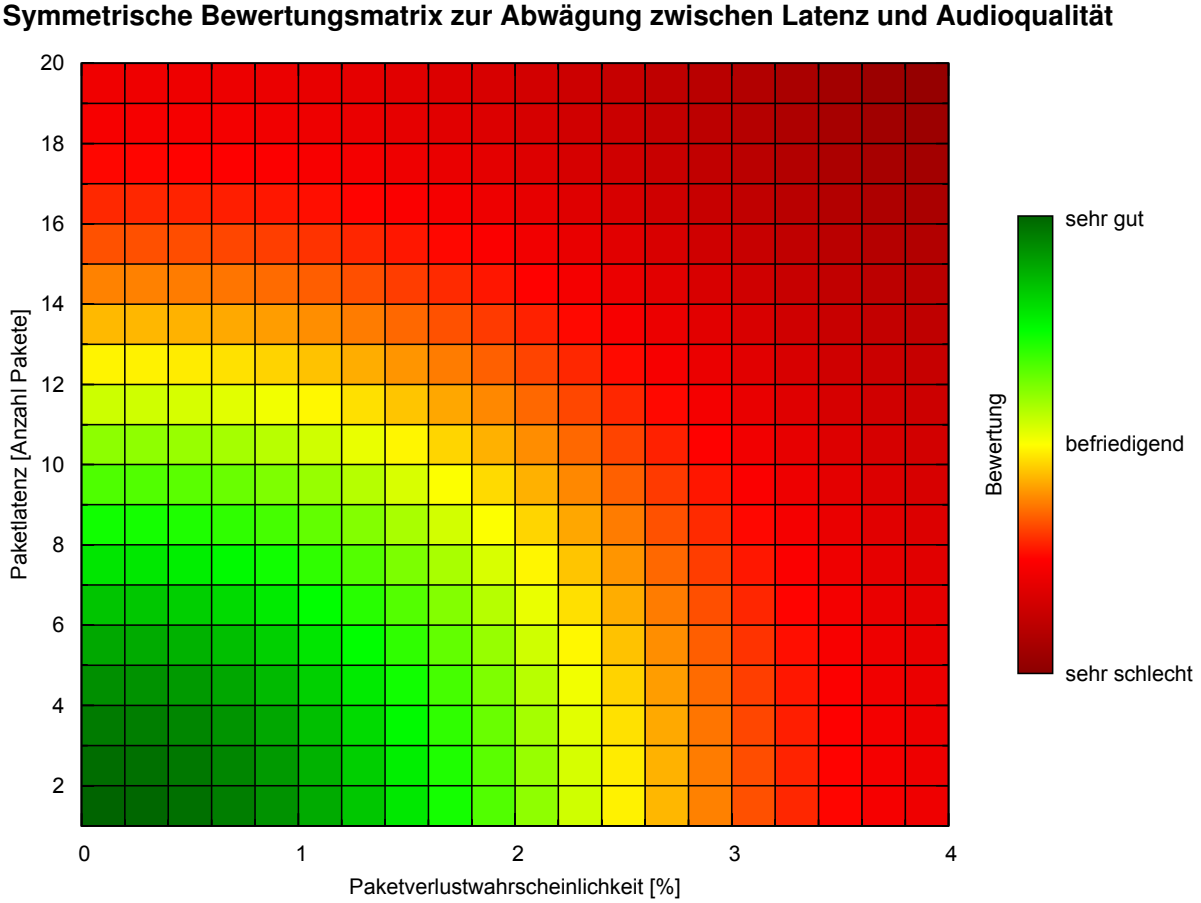

Abbildung 6.10: Symmetrische Zufriedenheitsmatrix

tiellen Längen des Abspielpuffers während des letzten Messintervalls wider.

Adaption durch Maximierung der Zufriedenheit

Unter der Annahme gleich bleibender Datenübertragungseigenschaften auch während der anschließenden Messperiode führt die Dimensionierung des Abspielpuffers mit dem Index i_{max} des Wertepaares mit der höchsten Bewertung $m_Z(i_{max}) \geq m_Z(i), i = \{1,\ldots,20\}$ zur Maximierung der Zufriedenheit.

Exemplarisch stellen wir diese Entscheidungsfindung an folgendem, in der Tabelle 6.5 dargestellten, Szenario dar. Die Berechnung erfolgt auf Basis eines gegebenen Histogramms H einer zurückliegenden Messperiode. Aus diesem wird ermittelt, dass die minimale Latenz sechs Pakete bzw. $6 \cdot t_\varphi = 18ms$ und die maximale 13 Pakete beträgt. Für diese Latenzen $k = m_L(6),\ldots,m_L(13)$ werden die korrespondierenden Paketverlustraten $m_Q(k)$ berechnet.

Mit der angenommenen Verteilung sehen wir, dass knapp 80% aller Pakete nach einer Latenz von $6 \cdot t_\varphi$ den Client erreichen. Mit einer zusätzlichen Wartezeit von einem t_φ würden noch immer 6.1% aller Pakete verspätet eintreffen. Dieser Anteil reduziert sich stetig, wobei für die angestrebte Paketverlustrate von unter 1% eine Abspielverzögerung von 11 Paketen vorzusehen ist.

Die Bestimmung des Wertepaares mit der potentiell höchsten Zufriedenheit erfolgt abschließend über ein Nachschlagen in der Bewertungsmatrix und der Ermittlung von $m_Z(k)$. Für die ersten beiden Wertepaare wird die Paketverlustrate auf 4% limitiert ($m_Z(0) = f_Z(6,0.04), m_Z(1) = f_Z(7,0.04)$), so dass in deren Berechnung nur der Faktor Latenz berücksichtigt wird. Für die übrigen Werte wird der Zufriedenheitswert in Abhängigkeit von beiden Präferenzparametern nachgeschlagen.

k	0	1	2	3	4	5	6	7
$m_L(k)$ [Pakete]	6	7	8	9	10	11	12	13
$m_Q(k)$ [%]	19.6	6.1	3.4	2.6	1.4	0.6	0.4	0
$m_Z(k)$ [%]	18.7	17.9	31.0	47.3	65.7	68.9	64.9	60.4

Tabelle 6.5: Exemplarisches Szenario für eine Zufriedenheitsbewertung

Die Zufriedenheit erreicht mit einer Abspielverzögerung von 11 Paketen bei eine Paketverlustrate von 0.6% mit der Bewertung von 68.9% ihr Maximum. Bei unabhängiger Wahlmöglichkeit würde eine solche Dimensionierung des Puffers statistisch betrachtet dem Musiker die bestmögliche Zufriedenheit gewähren.

In einer laufenden NMP Sitzung sind die Entscheidungen dagegen nicht unabhängig und wahlfrei möglich. Ganz naheliegend stellt jede Anpassung des Abspielpuffers eine Diskontinuität im Audiodatenstrom dar: wird er verkürzt, werden bereits zur Ausgabe eingereihte Pakete überschrieben, wird er verlängert, entstehen Lücken im Audiodatenstrom. Zwar können wir mit den vorgestellten Mechanismen zur Verschleierung von Paketverlusten die negativen Auswirkungen von Diskontinuitäten abmildern, komplett verhindern lassen sich Störungen dagegen nicht. In jedem Fall muss die Diskontinuität im Zuge der Adaption bei der Berechnung der Zufriedenheit durch eine Erhöhung der resultierenden Paketverlustrate berücksichtigt werden. Da uns das Messfenster n_H und die absolute Anzahl der Paketverluste im zurückliegenden Messzeitraum bekannt sind, kann die Berechnung der effektiven Paketverlustrate a priori erfolgen und eine Verfälschung der Zufriedenheit durch die Adaption ausgeschlossen werden.

Noch störender als die unvermeidbaren Störungen im Audiosignal nach einer Anpassung des Abspielpuffers ist für den Musiker die daraus resultierende Änderung der Systemlatenz. Das Prinzip von NMP verlangt dem Musiker die Fähigkeit ab, sich an eine gegebene Wahrnehmungslatenz anzupassen. Das gelingt bis zum anvisierten Wert von 30 ms gut und darüber hinaus mit Übung, allerdings muss die Latenz dabei konstant sein. Eine kontinuierlich schwankende Systemlatenz macht somit die Praktikabilität von NMP zunichte.

Daher können wir die Adaption nicht allein auf Basis der ermittelten Zufriedenheitswerte treffen, sondern müssen einen Adaptionsverlauf mit dem Ziel minimaler Anpassungsvorgänge anstreben. Für den Entwurf des Ablaufs wurden in typischen NMP Sitzungen gültigen Aspekte berücksichtigt und folgende Schlussfolgerungen geschlossen:

1. Aus der Forderung zeitlich stabiler und gleich bleibender Netzeigenschaften folgt implizit, dass deren Bestimmung bereits innerhalb der ersten Messperiode recht zuverlässig möglich ist.
2. Werden im Verlauf der Sitzung potentielle Verbesserungen der Zufriedenheit über eine Erhöhung des Abspielverzögerung erkannt, wird eine Anpassung vorgenommen.
3. Um Oszillationen zu vermeiden, wird im umgekehrten Fall, wenn also eine Verringerung der Abspielverzögerung eine höhere Zufriedenheit verspricht, die Anpassung erst bei Überschreiten einer vorgegebenen höheren Schwelle durchgeführt.

Diese asymmetrische Anpassung stellt sicher, dass schwankende Netzeigenschaften nicht zu kontinuierlichen Adaptionen führen. Die aufgeführte Präferenz hin zu einer Erhöhung der Abspielverzögerung basiert auf unseren Erfahrungen, dass die Anzahl der Anpassungsvorgänge hierbei geringer ist. Unabhängig davon kann eine Bevorzugung der Gegenrichtung konfiguriert werden. Essenziell ist dabei lediglich, eine signifikante Asymmetrie beizubehalten.

Mit dem Werkzeug der kontinuierlichen Erfassung von Histogrammen über die gemessenen Paketlatenzen haben wir diese Asymmetrie einfach und effizient umgesetzt. Wir verwenden dabei für die Entscheidungsfindung zwei Histogramme, die über unterschiedlich breite Messfenster ermittelt werden. Das kürzere

enthält dabei $n_{H_k} = 2 \cdot 375 = 700$ Messwerte für die Dauer von zwei Sekunden, das lange $n_{H_l} = 16 \cdot 375 = 5600$ Messungen über 16 Sekunden. Die Adaption erfolgt anschließend derart, dass für die Erhöhung der Abspielverzögerung die zurückliegende Paketverlustwahrscheinlichkeit aufgrund von H_k, während für eine Verringerung H_l herangezogen wird. Im Ergebnis können kurzfristige Verschlechterungen der Audioqualität eine Erhöhung der Latenz bewirken, die erst wieder zurückgenommen werden, wenn sie über längere Zeit auch bei geringerer Latenz zufriedenstellend wäre.

Mit diesem in unserem NMP-System umgesetzten Ansatz haben wir sehr gute Ergebnisse erzielt: meist wird bereits nach der ersten Messperiode die optimale Abspielverzögerung ermittelt, die für den Rest der Sitzung gültig bleibt. Bei signifikanten Änderungen der Netzeigenschaften erfolgt eine Anpassung sehr zeitnah und meist innerhalb von ein bis zwei Adaptionsvorgängen. In emulierten Netzen mit dynamischen Netzverhalten verhindert die asymmetrische Trägheit zuverlässig eine oszillierende Anpassung.

6.5.2.3 Personalisierung der Präferenzmatrix

In der beschriebenen Form erlaubt die Bewertungsmatrix größtmögliche Flexibilität bei der Anpassung an individuelle Bedürfnisse und Vorgaben. In der Praxis hat sich allerdings deren Umsetzung als sehr aufwändig und daher schwer durchführbar erwiesen. Für die oben vorgestellte Matrix mit 380 diskreten Feldern müsste ein Anwender bereits eine recht genaue Vorstellung seiner Präferenzen haben, für die er vorab jede potentielle Wertekombination testen und bewerten müsste. Neben der Abhängigkeit von subjektiven Empfindungen kommt erschwerend hinzu, dass der Stellparameter Audioqualität nur unzureichend evaluiert werden kann. Die Paketverlustrate während einer gegebenen Spieldauer hängt vom Inhalt der Audiodaten und von der Verteilung der Lücken im Datenstrom ab. So können in einem Testlauf viele Paketverluste genau in die Spielpausen fallen und somit praktisch unerkannt bleiben, während in einem zweiten Durchgang Diskontinuitäten auf signifikante Passagen fallen können und die wahrgenommene Audioqualität bei gleicher Paketverlustrate deutlich verringern.

Diese auch weiterhin verfügbare Möglichkeit, dem Anwender die vollständige Anpassung der Präferenzmatrix zu erlauben, haben wir um solche erweitert, mit denen die wesentlichen Eigenschaften einfach und nachvollziehbar konfiguriert werden können. Grundlage dafür sind die während unserer Arbeit mit Musikern gewonnenen Erkenntnisse, dass die mit der symmetrischen Basismatrix erzielten Bewertungen allgemeingültigen Charakter haben und nur um eine Möglichkeit zur unterschiedlichen Gewichtung von Latenz und Audioqualität erweitert werden mussten.

Mit der vorgestellten Konstruktion der Präferenzmatrix ist eine solche Erweiterung einfach möglich, indem wir den Anteil der Bewertungsgrößen am Zufriedenheitsindex über einen Gewichtungsparameter p variieren. Die resultierende Berechnung erfolgt dann über die Formel

$$f_{Z_p}(x,y,p) = \frac{\sqrt{f_Q(x)^{-(1-p)} \cdot f_L(y)^{-p}} + \sqrt{f_Q(x) + f_L(y)} \cdot 0.3}{1 + \sqrt{2} \cdot 0.3} \qquad \text{mit} \qquad 0 \le p < 1$$

p bewegt sich dabei zwischen den Extremwerten 0 und 1, die jeweils der alleinige Berücksichtigung von Latenz bzw. Qualität entsprechen. Mit einem Wert von $p = 0.5$ ist die Gewichtung symmetrisch, es gilt $f_{Z_p}(x,y,0.5) = f_Z(x,y)$. Die einfache Möglichkeit, über einen einzigen Wert die Gewichtung der Bewertung anzupassen, hat die Personalisierung der Matrix für Musiker attraktiv gemacht.

In Abbildung 6.11 sind die Präferenzmatrizen für $p = 0.8$ und $p = 0.2$ abgebildet. Mit der gewählten symmetrischen Verschiebung der Gewichtung um das Zentrum von 0.5 sind die Matrizen ebenfalls zur z-Achse rotationssymmetrisch.

Tabelle 6.6 erweitert das in Tabelle 6.5 angenommene Szenario um die Zufriedenheitswerte bei Verwendung der Latenz (m_{Z_l}) bzw. (m_{Z_q}) präferierenden Bewertungsmatrizen. Mit m_{Z_l} wird das Optimum von 30.8

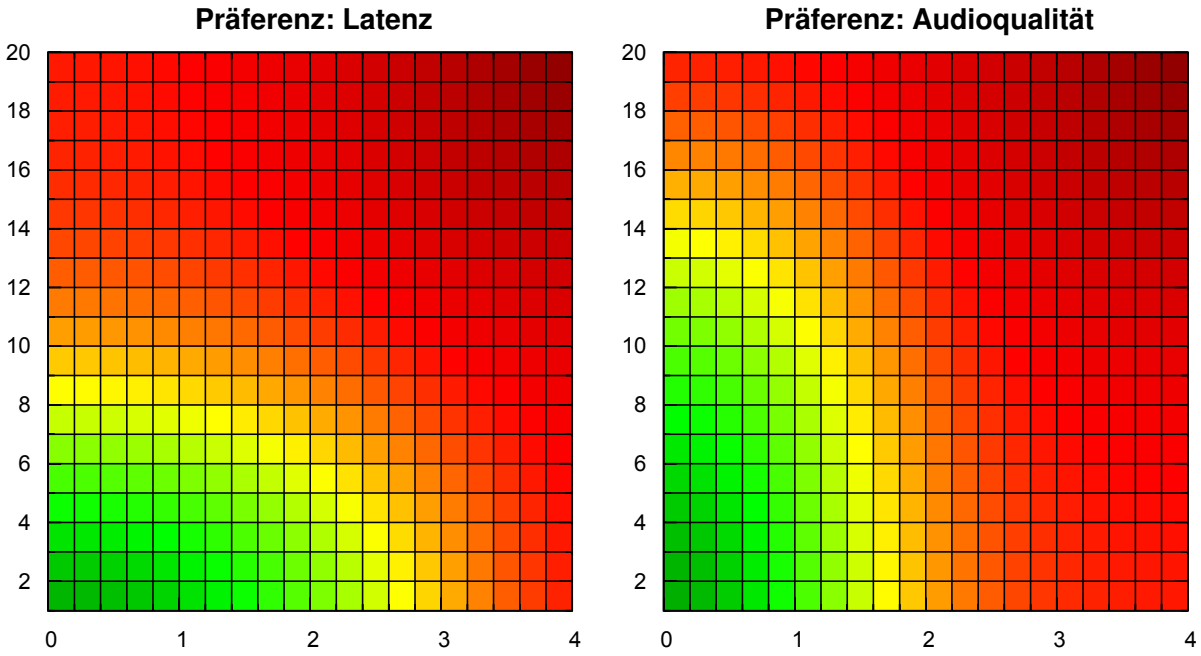

Abbildung 6.11: Personalisierte Zufriedenheitsmatrizen mit $p = 0.8$ und $p = 0.2$

bei einer Abspielverzögerung von 10 Paketen und 1.4% Paketverlustrate erreicht, wohingegen die Verwendung von m_{Z_q} eine Latenz von 12 Paketen präferiert, mit der die Paketverlustrate auf 0.4% reduziert wird.

So einfach und effektiv die Personalisierung der Zufriedenheitsmatrix über den Gewichtungsfaktor p ist – es verbleiben bestimmte Szenarien oder Präferenzen von Musikern, die mit einer angepassten Gewichtung nicht abgebildet werden können. Man denke dabei an Anwender, die unabhängig von der Paketverlustrate eine gleich bleibende Latenz verlangen bzw. sich nur an wenige Latenzen adaptieren können.

Die Berücksichtigung solcher quasi-statischen Bewertungen ist zwar einfacher als über die vorgestellte Bewertungsmatrix umsetzbar, erfordert allerdings einen separaten Betriebsmodus der Software, der die Präferenzen in diskrete Algorithmen abbildet. Von dieser Diversifizierung haben wir abgesehen und setzen für die Abwägung generell auf eine Bewertungsmatrix.

Die beschriebenen diskreten Präferenzen können jeweils mit Matrizen abgebildet werden, indem der gewünschte Verlauf der Anpassung direkt in die Felder eingetragen wird. Zur Veranschaulichung sind zwei solcher Matrizen in Abbildung 6.12 dargestellt.

Die linke Matrix bildet die Präferenz des Musikers ab, der mit einer konstanten Latenz von 10 Paketen arbeiten möchte. Sinnvolle Szenarien für einen solchen Fall kann das Musizieren einer Band sein, die bereits länger zusammen spielen und dabei die NMP Sitzungen über die immer gleichen Netzzugänge aufbauen. Dadurch sind die Netzeigenschaften relativ konstant und können mit einer konstanten Latenz bespielt wer-

k	0	1	2	3	4	5	6	7
$m_L(k)$ [Pakete]	6	7	8	9	10	11	12	13
$m_Q(k)$ [%]	19.6	6.1	3.4	2.6	1.4	0.6	0.4	0
$m_Z(k)$ [%]	18.7	17.9	31.0	47.3	65.7	68.9	64.9	60.4
$m_{Z_l}(k)$ [%]	18.6	17.8	21.6	26.2	30.8	29.5	26.6	24.7
$m_{Z_q}(k)$ [%]	18.6	17.8	17.5	19.3	36.0	68.1	69.7	68.9

Tabelle 6.6: Zufriedenheitsbewertung mit symmetrischer und personalisierten Matrizen

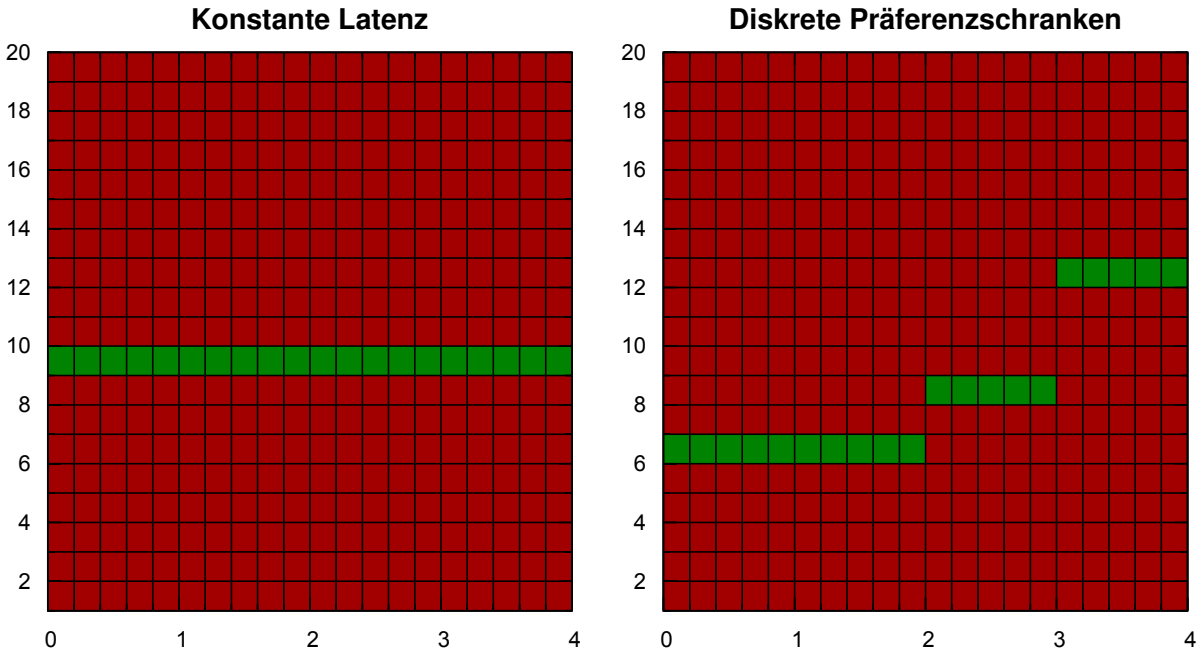

Abbildung 6.12: Diskrete Zufriedenheitsmatrizen

den. Das wiederum impliziert, dass bei Änderungen der Netzeigenschaften ein Betrieb des NMP-Systems mit der gewählten Latenz nicht mehr möglich ist und auch nicht automatisch angepasst wird.

Die Berechnung der Zufriedenheit über die entsprechende Matrix $m_{Z_{c(10)}}$ kann wie üblich verlaufen und liefert nur für Wertepaare mit $m_L(10)$ Zufriedenheitswerte größer Null.

Die rechte Matrix ist eine Erweiterung des beschriebenen Szenarios um zwei zusätzliche Bereiche gleich bleibender Latenzen. Die Musiker tolerieren innerhalb der ersten Zone bei einer Latenz von sechs Paketen Verlustraten von bis zu 2%. Höhere Verlustraten zwischen zwei und drei Prozent sollen mit einer Latenz von acht Paketen begegnet werden, während im letzten Abschnitt zwischen drei und vier Prozent Verlustrate eine Latenz von 12 Paketen verwendet werden soll.

Plausible Anwendungen für eine solche Matrix lassen sich ebenfalls als Erweiterung des ersten Szenarios ableiten. Man stelle sich die gleichen Musiker vor, die diesmal in verschiedenen Netzen zu NMP Sitzungen zusammen kommen. Im optimalen Fall erfolgt das über ein lokales Netz, bspw. jeweils über Clients, die am Campus-Netz einer Hochschule angebunden sind. Hier sind sehr zuverlässige Netzeigenschaften anzutreffen und ermöglichen einen verlustarmen Betrieb auch bei einer geringen Latenz. Klinken sich die Musiker dagegen aus ihren Studentenwohnungen in das System ein, verläuft der Zugang zum nahe gelegenen NMP-Server über einen kommerziellen Zugangsprovider, wodurch sich Latenz und Jitter erhöhen und ein zufriedenstellender Betrieb nur mit einer höheren Abspielverzögerung durchführbar ist. Kommen zuletzt die Musiker während der Semesterferien zu NMP Sitzungen zum weiter entfernt gelegenen Server zusammen, steigen Übertragungsverzögerungen und Jitter weiter und erfordern einen Betrieb mit noch höheren Abspielpuffern.

Mit der dargestellten Bewertungsmatrix erfolgt die Adaption derart, dass die Zone mit der geringsten Latenz gewählt wird, die einen Betrieb mit einer Paketverlustrate innerhalb der vorgesehenen Grenzen ermöglicht. Das impliziert, dass die Adaption nicht bei allen Teilnehmern einer Sitzung gleich sein muss. Ist bspw. ein Musiker über das Campusnetz mit dem NMP-Server verbunden und musiziert mit einem von zuhause angebundenen Mitspieler, kann er evtl. mit einer Latenz von sechs Paketen arbeiten, während sein Partner mit der doppelten Anzahl arbeiten muss, um eine erträgliche Paketverlustrate einzuhalten.

6.5.2.4 Anwendung der Adaption

Der generelle Ablauf eines Clients mit dem vorgestellten Verfahren zur adaptiven Anpassung der Abspielverzögerung ist pseudoalgorithmisch in Algorithmus 5 dargestellt. Zu jeder durch die Audiohardware initiierten Aktivierung des Clients werden die von der Netzhardware empfangenen Pakete eingelesen, dabei deren Alter bestimmt und dem Statistikmodul zugeführt.

Dieses berechnet nach dem in Algorithmus 4 vorgestellten Verfahren abschnittsweise Histogramme über die relativen Häufigkeiten der Paketlaufzeiten. Mit der darauf basierenden kumulativen Häufigkeit wird für jeden Messabschnitt für alle zu berücksichtigenden Abspielverzögerungen die korrespondierende Paketverlustrate berechnet, die als Vorhersage für den folgenden Abschnitt herangezogen wird.

Die zu den ermittelten Wertepaaren von Qualität zu Latenz korrespondierenden Zufriedenheitswerte werden in der gewählten Präferenzmatrix nachgeschlagen und abschließend als Eingabeparameter dem Adaptionsmodul zugeführt, welches die optimale Abspielverzögerung für den folgenden Abschnitt unter Einhaltung vorgegebener Einschränkungen ermittelt. Diese Berechnung wird für jede Messperiode einmal durchgeführt, die errechnete Abspielverzögerung bleibt im nachfolgenden Zyklus gleich.

Die Sequenznummer des bei jeder Aktivierung an die Audiohardware auszugebenden Datenblocks wird berechnet aus der Differenz der Sequenznummer des zuletzt von dort ausgelesenen Blockes und der Abspielverzögerung.

Algorithmus 5 : Adaptive Abspielverzögerung am Client

```
Input : Aufruf innerhalb der ISR
inBlock = audioIO.readBlock()          /* lies Block von Audiohardware ein */
sendPacket = createPacket(inBlock, packetSeq)
netIF.sendPacket(sendPacket)            /* versende Audiopaket(readSeq) */
while netIF.hasData() do
    /* bearbeite alle Pakete im Empfangspuffer der Netzkarte                  */

    recvPacket = netIF.getPacket()
    playoutBuffer.addPacket(recvPacket)
    packetLatency = packetSeq- sequenceNumber(recvPacket)
    H.add(packetLatency)
end while
if H.hasUpdated() then
    for latency =1 to 20 do
        quality = H.calculateQualityAtLatency(latency)
        m_Z[latency ]=satisfactionMatrix.lookup(latency, quality)
    end for
    playoutDelay = findOptimalDelay(playoutDelay, m_Z)
end if

/* lese Paket mit berechneter Abspielverzögerung                          */

outBlock = playoutBuffer.getPacket(packetSeq-playoutDelay)
audioIO.writeBlock(outBlock) /* und schreibe es in die Audiohardware       */

packetSeq ++
```

6.5.3 Bewertung und Fazit

Wir haben in diesem Abschnitt zwei unterschiedliche Varianten für die Bestimmung der Abspielverzögerung vorgestellt.

Das statische Verfahren genügt der zentralen Forderung an unser NMP-System, eine konstante Latenz über die gesamte Dauer einer Sitzung einzuhalten. Diese ist mit der Vorgabe, dass Daten ihre zugewiesenen Sequenznummern über die gesamte Lebenszeit beibehalten, sehr einfach zu realisieren. Der bei jeder Aktivierung des Clients von der Audiohardware gelesene Datenblock wird mit einer fortlaufenden Sequenznummer versehen, woraus sich die Sequenznummer des in diesem Zyklus auszugebenden Paketes implizit aus der Differenz dieses Wertes und der vorgesehenen Latenz berechnen lässt. Wir haben gezeigt, dass bei einer Zykluszeit von $t_\varphi = 2.66\,ms$ eine konstante Systemlatenz von knapp unter 30 ms bei einer Abspielverzögerung von acht Paketen erreicht wird.

Neben dem praktischen Vorteil einer einfachen Umsetzbarkeit ist dieses statischen Verfahren für den Anwender durch die durchgehend gleichbleibende Systemlatenz interessant, da die Anforderungen an die persönliche Adaptionsfähigkeit dadurch verringert werden. Dieser Vorteil ist gleichzeitig auch sein Nachteil, denn wir haben gezeigt, dass eine statisch gewählte Abspielverzögerung selten optimal ist.

Der adaptive Ansatz ist in der Lage, die Abspielverzögerung an die während einer Sitzung herrschenden Netzeigenschaften anzupassen. Grundidee dabei ist, die Paketlaufzeiten innerhalb vorgegebener Messabschnitte statistisch auszuwerten und daraus Rückschlüsse auf die Abhängigkeit von Audioqualität zu Latenz zu schließen. So kann mit den vorgestellten Mechanismen für jeden abgelaufenen Zyklus eine Aussage darüber getroffen werden, welche Paketverlustrate bei der Verwendung eines Abspielpuffers mit n Paketen erzielt werden hätte können. Mit den für einen relevanten Bereich – bei uns bspw. von eins bis 20 Paketen – errechneten Wertepaare kann anschließend für die Bestimmung der Abspielverzögerungen eine Abwägung zwischen Latenz und Qualität getroffen werden.

Für eine solche subjektive Entscheidung haben wir das Konzept der personalisierbaren Präferenzmatrizen eingeführt, die dem Anwender die Freiheit gewährt, jede Kombination aus Latenz und Audioqualität mit einem Zufriedenheitswert zu bewerten. Die in einem Messabschnitt berechneten Werte für diese Abhängigkeit werden in der Präferenzmatrix nachgeschlagen, womit eine Aussage darüber getroffen werden kann, mit welcher Latenz und der damit erzielten Audioqualität er momentan am zufriedensten wäre. Ein dediziertes Adaptionsmodul berechnet abschließend anhand der Zufriedenheitswerte die fortan zu verwendende Abspielverzögerung unter Berücksichtigung limitierender Vorgaben (wie Minimierung der Anpassungsvorgänge oder Verhinderung von Oszillationen).

Um den Aufwand für die Erstellung einer Präferenzmatrix zu minimieren, haben wir auf Grundlage allgemeingültiger Bewertungen und empirischen Werten eine normierte Basismatrix konstruiert. Über einen Gewichtungsparameter kann diese den Vorgaben des Anwenders entsprechend über die Präferenz zwischen Latenz und Qualität justiert werden. Darüber hinaus können diskrete Verläufe ebenso mit der Präferenzmatrix abgebildet werden, einschließlich der Verwendung einer konstanten Latenz.

Das adaptive Verfahren liefert für gleich bleibende Netzeigenschaften nach spätestens zwei Messzyklen zuverlässig den von einem Nutzer präferierten Wert für Latenz und dazugehöriger Audioqualität. Bei Änderungen der Netzeigenschaften kann über eine Anpassung der Abspielverzögerung eine Verbesserung der momentanen Zufriedenheit erreicht werden. Da diese Änderung allerdings auch eine Anpassung der Latenzkompensation beim Musiker erfordert, kann sie absolut zu einer geringeren Gesamtzufriedenheit führen.

Aus den Untersuchungen mit Musikern hat sich während unserer Arbeit die Präferenz für eine kombinierte Strategie herauskristallisiert: am zufriedensten sind sie, wenn in einer initialen Einspielphase das adaptive Verfahren angewendet wird, um eine gute Balance zwischen Latenz und Qualität zu bestimmen.

In der folgenden Phase sollte es dagegen keine Anpassungen mehr geben. Diese Vorgaben sind in das Adaptionsmodul geflossen, welches eigenständig die Sitzung in Adaptions- und statischer Phase unterteilt und spätere Anpassungen der Abspielverzögerung nur unter großen Einschränkungen veranlasst.

Damit entspricht unser adaptives Verfahren den Anforderungen der Musiker und ist in der Lage, für die geltenden Netzeigenschaften eine optimale Zufriedenheit zu gewähren. Musiker, die unabhängig von den Netzeigenschaften immer mit einer definierten Latenz arbeiten möchten, können dagegen auf die Adaption verzichten und entweder das statische Verfahren oder das adaptive mit einer entsprechend diskreten Präferenzmatrix verwenden.

6.6 Zeitsynchronisation

Gemeinsames Musizieren impliziert Gleichzeitigkeit und stellt damit eine sehr strikte Anforderung an die Synchronisierung der teilnehmenden Clients untereinander dar. Mit diesem Problem muss sich jede verteilte Anwendung auseinandersetzen, die interagierenden Teilnehmern oder Systemen eine gemeinsame Zeitbasis bereitstellen muss. Die für NMP erforderliche hohe Genauigkeit gepaart mit der großen Anzahl gleichzeitiger Teilnehmer schränkt die Anwendbarkeit heute geläufiger Maßnahmen ein und erfordert eine besondere Betrachtung der Synchronität, der wir uns in diesem Abschnitt widmen. Wir beginnen mit einer Problembeschreibung und stellen etablierte Lösungsverfahren vor. Anschließend diskutieren wir die Einschränkungen und Besonderheiten bei NMP und beschreiben das von uns eingesetzte Verfahren.

6.6.1 Überblick

Die Erfordernis von Maßnahmen zur Synchronisierung hat ihren Ursprung in der Gangabweichung der in einer Anwendung verwendeten Taktgeber. Diese Bauteile (meist in Form von Quarz-Oszillatoren) versorgen Einrichtungen und Geräte mit einer nominellen Taktrate. Technische Einschränkungen führen dazu, dass dieser nominelle Takt f_{nom} immer nur mit einer Abweichung zu der realen Frequenz $f_{real} = f_{nom} + f_{\Delta}$ angenähert wird. Die relative Ungenauigkeit $G_f = \frac{f_{\Delta}}{f_{nom}}$ bestimmt dabei die Güte eines Taktgebers, die sich in einem ein- bis dreistelligen *ppm* (parts per million) Bereich ($G_f = [0.000005 \ldots 0.0003]$) bewegt. Muss der Arbeitstakt einer Anwendung indirekt als Teil oder Vielfaches der Oszillatorfrequenz abgeleitet werden, können die Abweichungen aufgrund von Rundungsfehlern bis in den Prozentbereich hineinreichen.

Erschwerend kommt hinzu, dass die Genauigkeit nicht konstant ist. Vielmehr variiert sie aufgrund physikalischer Gegebenheiten (wie Betriebstemperatur oder Luftfeuchtigkeit) oder durch technische Einflüsse. Geringe Gangabweichungen und damit höhere Genauigkeiten erfordern eine besondere Auslese der Taktgeber einerseits und die Minimierung von Umwelteinflüssen auf die Abweichung andererseits. Dafür sind bisweilen umfangreiche technische Vorkehrungen zu treffen (bspw. Heiz- und Kühlvorrichtungen zur Sicherstellung einer konstanten Betriebstemperatur) und beschränken den Einsatz sehr genauer Taktgeber auf hochpreisige Produkte.

Abseits der technischen Hintergründe ist Asynchronität ein alltägliches Phänomen. Liebhaber antiker Uhren werden das Nachstellen der Uhrzeit als Teil ihres Hobbys ansehen, wohingegen der Amateurfilmer nur einmal den Fehler macht, Video und Audio mit verschiedenen und voneinander unabhängigen Geräten aufzunehmen. Denn diese werden mit hoher Wahrscheinlichkeit aufgrund ungleicher Gangunterschiede beider Taktgeber nicht ohne Weiteres synchronisierbar sein.

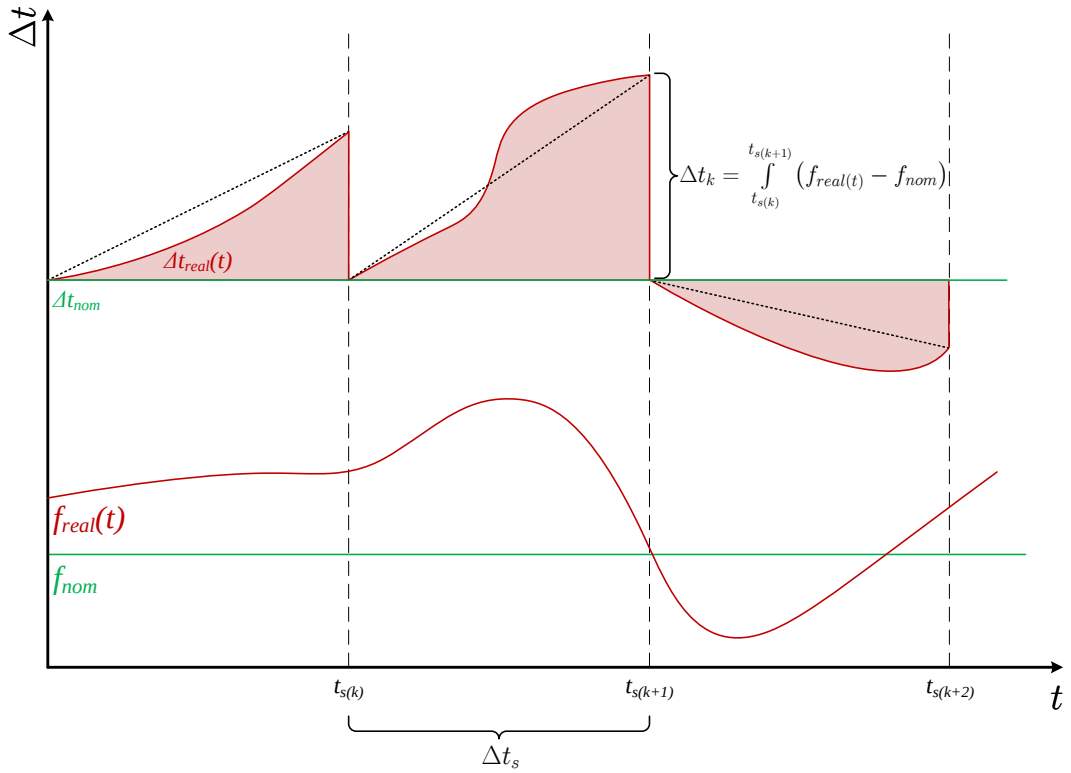

Abbildung 6.13: Prinzip der Zeitsynchronisierung

6.6.2 Takt- und Zeitsynchronität

Diese Beispiele eignen sich trotz ihrer Banalität zur Verdeutlichung zweier grundlegender Formen der Synchronisierung: zum einen gibt es die Zeit-Synchronisierung von Systemen an ein definiertes Zeitnormal. Dass die Tagesschau genau um acht beginnt ist nur unter der Annahme richtig, dass eine gemeinsame und durchweg akzeptierte Zeitreferenz existiert – bspw. die koordinierte Weltzeit *UTC* (Universal Time Coordinated), die auch von der PTB in Braunschweig bereitgestellt wird. Die Zeit-Synchronisation wird für eine periodische Anpassung der eigenen Uhr an ein absolutes Zeitnormal eingesetzt, mit dem Ziel, die mittlere Abweichung der lokalen Systemzeit vom Zeitnormal zu minimieren.

Bei der Takt-Synchronisation hingegen liegt der Fokus auf der Gleichtaktung von verschiedenen Systemen innerhalb einer Anwendung relativ zueinander. Wie im obigen Beispiel der Videokamera ist primär wichtig, dass Bild und Ton nicht auseinanderdriften, während die absolute Genauigkeit der Zeitmessung meinst zweitrangig ist. So muss die Kamera ein Ton-Bild-synchrones Video liefern, welches meist nicht zwingend die nominelle Bildwiederholfrequenz von 25 Hz hat und dadurch beim Abspielen auf einem anderen Gerät schneller oder langsamer läuft. (Anmerkung: dieses geschieht tatsächlich immer, jedoch ist es für den Betrachter subjektiv nicht unterscheidbar, ob ein Film mit 24.9 oder 25.1 Hz abgespielt wird. Misst man dagegen die Abspieldauer auf verschiedenen Geräten mit einer Stoppuhr, können Laufzeitunterschiede von mehreren Sekunden pro Stunde gemessen werden.)

Offenkundig sind die beiden Synchronisationsvarianten voneinander abhängig: ein Takt-synchronisiertes System ist implizit lokal Zeit-synchronisiert. Umgekehrt kann bei hinreichend kurzen Synchronisationsintervallen ein Taktabgleich mit Hilfe der Zeitsynchronisation angenähert werden.

Das Prinzip der Zeitsynchronisation ist in Abbildung 6.13 dargestellt. Von der nominalen Taktfrequenz

f_{nom} variiert der reale Takt f_{real} mit der Zeit t. Die Synchronisation erfolge in konstanten Intervallen von Δt_s zu den Zeitpunkten $t_{s(n)}$, zu denen die interne Uhr mit der Referenz abgeglichen und die Abweichung Δt auf Null zurückgesetzt wird. Zwischen zwei Synchronisationspunkten $t_{s(k)}$ und $t_{s(k+1)}$ verändert sich Δt kontinuierlich um die Differenz zwischen f_{real} und f_{nom}. Der Betrag der Abweichung unmittelbar vor der Synchronisation ergibt sich daher zu $\Delta t_{(k)} = \int_{t_{s(k)}}^{t_{s(k+1)}} \left(f_{real}(t) - f_{nom}\right) dt$.

In der Praxis ist die Variabilität des realen Taktes gering und meist einseitig, d.h., das Vorzeichen der Abweichung bleibt über relativ lange Zeitabstände gleich, so dass Δt innerhalb eines Synchronisationsintervalls monoton ist und im Ergebnis die Abweichung unmittelbar vor der Synchronisation einen lokalen Extremwert erreicht. Der in Abbildung 6.13 im letzten Drittel dargestellte Verlauf von $f_{nom}(t)$ mit einem Übergang von negativer zu positiver Taktabweichung ist ein der Vollständigkeit halber aufgeführter und nur relativ selten auftretender Fall.

Der Umstand, dass die Abweichung zwischen zwei Synchronisationspunkten kontinuierlich anwächst, bedeutet im Umkehrschluss, dass die mittlere Abweichung direkt von der Größe des Synchronisationsintervalls abhängt. Es gilt somit $\Delta t_{(k)} \sim \Delta t_s$, woraus unmittelbar folgt, dass eine Minimierung der Abweichung durch eine Minimierung der Synchronisationsabstände erreicht wird.

Das ist aus Alltagserfahrungen intuitiv nachvollziehbar, denn je öfter eine Uhr nachgestellt wird, desto geringer fällt der zu korrigierende Betrag aus.

Relevant für die Unterscheidung der hier diskutierten Synchronisationsarten wird es, wenn die Anpassfrequenz $f_s = \frac{1}{\Delta t_s}$ in die Größenordnung der zu kontrollierenden Realfrequenz bewegt wird. Ist man in der Lage, eine Anpassung mit $f_s = f_{real}$ vorzunehmen, ist damit auch inhärent eine Taktsynchronität sicher gestellt. Wir können daher die beiden beschriebenen Synchronisationsverfahren anhand des Quotienten Q_{sn} aus Real- und Anpassfrequenz unterscheiden: für eine Takt-Synchronisierung muss $Q_{sn} = \frac{f_s}{f_{real}} \geq 1$ erfüllt sein, während bei Zeit-Synchronisation meist gilt $Q_{sn} \ll 1$.

Notwendige Bedingungen für die praktische Umsetzung von Taktsynchronisation sind nur in geschlossenen Systemen gegeben, in denen entweder ein gemeinsamer Takt verwendet wird oder voneinander abhängige Taktgeber über dedizierte Signale synchronisiert werden. Als Beispiel für ein solch geschlossenes System kann die bereits erwähnte Videokamera genannt werden, in der der Pixeltakt für das Video und die Abtastrate für das Audio über unterschiedliche Teilverhältnisse aus einem gemeinsamen Referenztakt generiert werden. Die Synchronisation der so erzeugten Einzeltakte wird anschließend über eine Phasenregelschleife (*PLL*, Phase Lock Loop) über die kontinuierliche Minimierung der Phasen zueinander erreicht.

In verteilten Systemen, in denen Signallaufzeiten eine Takt-Synchronisation verhindern, kann dagegen diese nur indirekt über den Zeit-Abgleich erreicht werden. Für NMP ist somit nur diese relevant, so dass wir an dieser Stelle die Unterscheidung der beiden erörterten Synchronisationsarten einstellen. Im Weiteren ist daher, wenn nicht explizit vermerkt, mit dem Begriff Synchronisation die auf Zeit-Abgleich basierende Variante gemeint.

Die Grundlagen der heute verwendeten Verfahren zur Zeit-Synchronisation werden folgend kurz vorgestellt.

6.6.3 Etablierte Verfahren zur Zeit-Synchronisation

Weltweit gültiges Zeitnormal ist heute die Internationale Atomzeit *TAI* (Temps Atomique International), die von global operierenden Zeitinstituten (wie der PTB) unter Verwendung vieler Atomuhren (derzeit über 250) ermittelt wird [35]. Von dieser ist die UTC abgeleitet, die im praktischen Einsatz breitere Verwendung findet. Für die Übermittlung dieses Zeitnormals an Endgeräte weltweit haben sich unterschiedliche Verfahren etabliert.

6.6.3.1 Synchronisation über DCF77

In Westeuropa ist in Alltagsanwendungen die Synchronisation über den Zeitsignalsender *DCF77* weitverbreitet, über den funkgesteuerte Uhren mit der genauen Zeit versorgt werden. Das Zeitsignal wird von der PTB (Physikalisch-Technischen Bundesanstalt) in Braunschweig bereitgestellt und als Zeitzeichen über Langwelle bei 77.5 kHz ausgestrahlt [70]. Der Empfang des Signals ist technisch wenig anspruchsvoll mit einer einfachen Ferritantenne zuverlässig realisierbar, wodurch DCF77-Empfänger in Funkuhren und -Weckern heutzutage preisgünstige Alltagsgegenstände geworden sind. Prinzipiell eignet sich das Verfahren auch für die Zeit-Synchronisation von Computern und verteilten Systemen, für welches vor der Etablierung alternativer Methoden entsprechende Einsteckkarten und Adapter eingesetzt wurden. Neben der im Vergleich zu modernen Verfahren geringen Genauigkeit im Bereich von 100 ms ist ihre Variabilität in Abhängigkeit von der Entfernung zwischen Sender und Empfänger eine wesentliche Schwäche der Synchronisation über DCF77.

6.6.3.2 Synchronisation über GPS

Basis für die Satelliten gestützte Positionsbestimmung über Triangulation beim *GPS* (Global Positioning System) [38] sind die Laufzeitunterschiede von Signalen zwischen verschiedenen Satelliten zum Empfänger. Prinzipiell wird dabei für jede Messung die Übertragungszeit des GPS-Signals von mindestens drei Satelliten zum Endgerät gemessen. Mit der ermittelten Signallaufzeit und bekannter Ausbreitungsgeschwindigkeit werden zunächst die Entfernungen zu den Satelliten bestimmt. Da die Position aller Satelliten im Orbit zu jedem Zeitpunkt allen GPS-Empfänger bekannt ist, ist eine Positionsbestimmung über Triangulation möglich. Die Genauigkeit der Laufzeitmessung beeinflusst damit direkt die Positionsgenauigkeit und macht die Zeitsynchronität zwischen Satelliten und Empfänger zum entscheidenden Faktor für die Güte der Messung.

Die GPS-Satelliten sind mit Atomuhren ausgestattet, die nach [67] bei einer Ganggenauigkeit von über $2 \cdot 10^{-13}$ für den praktischen Einsatz als gleich getaktet angenommen werden können. Bei der Synchronisierung des Empfängers bedient man sich der Überbestimmung, um die Gangabweichung beim Empfänger zu kompensieren. Dabei wird bei jeder Messung zunächst eine Ungenauigkeit bei der Zeitmessung am Empfänger berücksichtigt, so dass sich eine von der Abweichung abhängige Kurve möglicher Positionen errechnen lässt. Mit einer Abstandsmessung zu einem weiteren vierten Satelliten kann dann, unter der Annahme einer in allen Einzelmessungen gleich eingeflossenen Zeitabweichung, die Asynchronität ermittelt und die genaue Position bestimmt werden. GPS-Empfänger werden damit bei jeder Messung an die UTC synchronisiert. Für eine Positionsbestimmung mit einer Genauigkeit von zehn Metern ist dabei eine Gangabweichung von unter 35 ns erforderlich.

Das systembedingte Vorhandensein einer sehr genauen Zeit macht so den Einsatz von GPS-Empfängern neben der Positions- auch zur Zeitbestimmung möglich. Einschließlich der Uhrendrift zwischen den Messungen kann für ein Endgerät eine weltweite Synchronisation mit Sub-Mikrosekunden Genauigkeit erreicht werden.

Die immense Nachfrage nach GPS-Empfängern für Navigationsgeräte hat in jüngerer Zeit zu einem Preisverfall geführt und die breite und kostengünstige Verfügbarkeit von Empfängern – auch für die 'missbräuchliche' Verwendung für die Zeitsynchronisation – ermöglicht. Ihre Anwendung als Uhrenbaustein in Rechnern verhindert allein die Anforderung, eine Sichtbarkeit mit mindestens vier GPS-Satelliten sicher zu stellen. Der so zu synchronisierende Rechner muss also draußen stehen oder mit einer externen GPS-Antenne verbunden werden, für Arbeitsplatzrechner beides keine praktikablen Ansätze. Da liegt es nahe, dedizierte Rechner mit solchen Möglichkeiten zu versehen und an diese *Zeitserver* verbundene Clients zu synchronisieren.

6.6.3.3 Zeitabgleich über NTP

Der weitläufige Bedarf einer Zeitsynchronisation in Netzen und verteilten Systemen zusammen mit der Erkenntnis, nicht jedes System mit hochgenauen Zeitgebern ausstatten zu können, hat bereits im Jahre 1985 zum Entwurf von *NTP* (Network Time Protocol [61]) geführt, einem Protokoll für die Synchronisation von Uhren in Computernetzen an die UTC.

Grundlage des verwendeten Verfahrens ist die Bestimmung der Zeitabweichung der eigenen Rechneruhr im Vergleich zu der eines NTP-Servers, die aus der Übertragungszeit von UDP-Paketen vom eigenen System zum NTP-Rechner und zurück berechnet wird. Stark vereinfacht lässt sich die Funktionsweise wie folgt beschreiben:

Grundannahme des Verfahrens ist, dass ein für die Messung versendetes Datenpaket den selben Weg in Hin- und Rückrichtung nimmt und dadurch die Übertragung jeweils gleich lange dauert. Der zu synchronisierende Rechner versieht das Datenpaket zum Startzeitpunkt mit seinem Zeitstempel und sendet es zum NTP-Server. Dieser fügt Zeitstempel für Empfangs- und Sendezeitpunkt hinzu und reflektiert das Paket zum Sender. Zusammen mit dem Empfangszeitpunkt kann aus diesen vier Zeitstempeln anschließend die Differenz der eigenen zur NTP-Uhr bestimmt und angepasst werden.

Da nicht alle Endsysteme weltweit von einem NTP-Server bedient werden können, erfolgt die Verteilung der UTC dabei hierarchisch in Strata: das Stratum 0 ist das Zeitnormal, das von Atomuhren bereitgestellt wird. Diese wird an direkt angebundene NTP-Server verteilt, die das Stratum 1 bilden, an die wiederum die Server des Stratums 2 angebunden sind. Diese Aufteilung kann noch mehrere Stufen beinhalten, bevor sie den zu synchronisierenden Client erreicht, wobei mit jeder zusätzlichen Stufe die Genauigkeit abnimmt.

An dieser Grundidee hat sich praktisch auch in den aktuellen Versionen *NTPv3* [59] bzw. dem leichter anwendbaren *SNTP* (Simple NTP [60]) nichts geändert. Hinzugekommen sind Verfeinerungen, um die mittlere Genauigkeit zu erhöhen. So wird bspw. bei jeder periodischen Anpassung nicht nur die Abweichung der Uhr angepasst, sondern die Gangabweichung des Taktgebers ermittelt und kontinuierlich korrigiert. Daneben wurden Techniken etabliert, die Ursachen von Ungenauigkeiten im Verfahren selbst detektieren und minimieren. Mit diesen Verfeinerungen lassen sich bereits relativ hohe Genauigkeiten erzielen: laut [62] können in lokalen Netzen unter idealen Bedingungen Synchronisationsgenauigkeiten von $200\mu s$ erreicht werden, während die Genauigkeit im Internet von zu vielen Faktoren abhängt und so nur ein Richtwert von 10 ms angegeben wird.

Eine tiefgehende Behandlung der Thematik grundlegender Zeitmessung und Synchronisation ist in [7] enthalten, umfassende Untersuchungen und Erweiterungen sind Bestandteile zahlreicher RFCs und Fachartikel. Für unsere Anwendung hat das Thema im Projektverlauf an Relevanz verloren, weswegen es hier nicht weiter vertieft wird.

Wir haben zwar im anfänglichen Stadium bei der Arbeit im Institutsnetz über NTP die Uhren von NMP-Server und –Clients angeglichen, waren uns jedoch auch bewusst, dass die Genauigkeit außerhalb des LANs nicht reichen wird, um Audiopakete mit Längen von 2.66 ms zu synchronisieren. Viel entscheidender für den mittlerweile erfolgten Verzicht auf jegliche Zeitsynchronisation zwischen den Endsystemen einer NMP-Sitzung ist die Einsicht, dass nicht die Gangabweichung der Rechneruhren problematisch ist, sondern die der Taktgeber in der Audiohardware. Dieser Problematik widmen wir uns im kommenden Abschnitt genauer und behandeln anschließend unseren Lösungsansatz einer Zeitreferenz-losen Synchronisierung der Audiodaten.

6.7 Audiodatensynchronisation im Server

Nach diesem Überblick über die allgemeine Synchronisationsproblematik richten wir unseren Blick in diesem Abschnitt auf ihre konkrete Auswirkung auf NMP und stellen das von uns verwendete Lösungsverfahren vor. Wir betrachten dabei zunächst die Situation bei realer Musik und lokalisieren die für eine praktische Anwendung erforderlichen Mechanismen zur Synchronisierung.

Anschließend betrachten wir die Schwierigkeiten, die sich in verteilten Audioanwendungen aufgrund von Jitter und Gangunterschieden der beteiligten Taktgeber ergeben, und gehen kurz auf etablierte Verfahren zur Kompensation ein. Die Besonderheiten, die NMP von anderen Audioanwendungen unterscheiden und welche Einschränkungen auf die anwendbaren Verfahren sich daraus ergeben, werden abschließend gesondert dargestellt.

Daraus leiten wir die Anforderungen für anzuwendende Synchronisationsverfahren in NMP ab und stellen den von uns eingesetzten Ansatz ausführlich dar.

6.7.1 Synchronitätsproblematik in der Musik

Das Musizieren ist von der Synchronisationsproblematik unmittelbar dann betroffen, sobald das Audiosignal digitalisiert und einem Takt zugeordnet wird – heute praktisch immer, da Aufzeichnung oder Effektbearbeitung in der digitalen Domäne erfolgen und eine zeitliche und räumliche Quantisierung erfordern. Gangabweichungen in einer Größenordnung von Sekunden pro Stunde sind nicht nur störend – sie machen Musizieren nicht praktizierbar.

Betrachten wir zur quantitativen Veranschaulichung beispielhaft die Güteklassen von Taktgebern digitaler Musikgeräte für eine Consumeranwendung des Hobbymusikers und der einer höherwertigen Ausstattung eines Berufsmusikers, die beide mit einer Abtastrate von 48 kHz arbeiten: in der Audiohardware des Berufsmusikers werden hochwertige Oszillatorquarze mit einer nominellen Frequenz gleich der Abtastrate von 48 kHz verwendet, wobei eine Gangabweichung von 30 ppm angenommen wird. Im Rechner des Hobbymusikers wird die Abtastrate für die Onboard Soundkarte aus dem CPU-Takt von 33 MHz gewonnen. Die nominelle Abtastrate kann entweder mit einem Teiler von 687 oder 688 angenähert werden, was zu einer Abweichung von 35 Hz bzw. 730 ppm führt.

Unter diesen Annahmen beträgt die potentielle Abweichung beim Berufsmusiker 0.18 Sekunden bzw. 8640 Samples pro Stunde, beim Hobbymusiker können sich die Gangabweichungen nach einer Stunde gar auf bis auf 2.63 s bzw. 126200 Samples aufsummieren. Eine voneinander unabhängige digitale Aufzeichnung der Tonspuren beider Musiker kann anschließend nicht mehr auf Sample-Basis miteinander gemischt werden.

Die in der Musik durchweg praktizierte Lösung ist eine Takt-Synchronisation. Zu diesem Zweck verfügt jedes professionelle Musikgerät über eine Global- oder World-Clock Schnittstelle, an die ein externer Zeitgeber angeschlossen wird, mit dem sich der interne Takt gleichschaltet. Es existieren unterschiedliche und miteinander konkurrierende Verfahren von Herstellern oder Vereinigungen, die hier nicht explizit erläutert werden. Allen gemein ist jedoch, dass sie auf eine dedizierte und drahtgebundene Übermittlung des Synchronisationssignals zwischen den Endgeräten aufsetzen. Die Synchronisationsperiode f_s ist dabei immer hinreichend klein, um einen Sample genauen Taktabgleich aller beteiligten Geräte zu gewährleisten.

Während Übungen und bei Live-Auftritten, bei denen es lediglich darum geht, die Sample-Synchronität einzuhalten, ist es ausreichend, ein Instrument oder Musikgerät als Takt gebenden Master zu verwenden. Damit darüber hinaus ein 74-minütiges Musikstück auch tatsächlich genau auf eine Audio-CD passt, muss bei Aufnahme und Produktion eine hohe absolute Ganggenauigkeit eingehalten werden, die durch den Einsatz

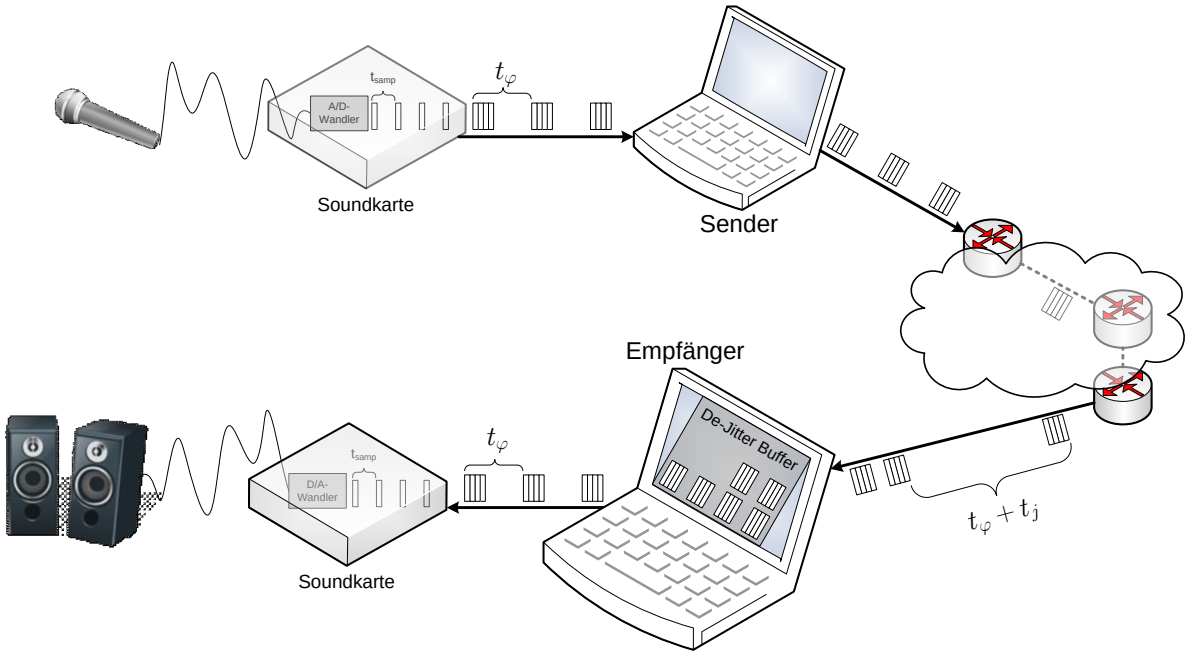

Abbildung 6.14: Streaming-Anwendung als Erzeuger-Verbraucher System

dedizierter und hochgenauer Taktgeber erreicht wird.

6.7.2 Synchronisationsverfahren in verteilten Audioanwendungen

Die einfache und effektive Methode zur Synchronisation über einen drahtgebundenen World-Clock lässt sich in verteilten Systemen nicht anwenden. Die einleitend beschriebenen Verfahren zur Zeitsynchronisation eignen sich zwar durchaus zum Abgleich von Rechneruhren innerhalb vorgegebener Genauigkeitsgrenzen. Diese wird allerdings unmittelbar dann wirkungslos, wenn Datenverarbeitung mit autonomen und von der Rechneruhr unabhängigen Taktgebern erfolgt. In der Praxis betrifft das nahezu alle Peripherie und somit auch die für NMP relevante Audiohardware.

Was für eine problemfreie Interaktion erforderlich wäre, ist eine Takt-Synchronisation der Taktgeber in der Audiohardware aller Teilnehmer einer NMP-Sitzung. Eine hinreichend hohe Genauigkeit im μs-Bereich ist dabei mit NTP nicht realisierbar. Ausreichend genau wäre zum heutigen Stand eine Synchronisation über GPS – im Ergebnis wird ein drahtloser Ersatz für das World-Clock Signal benötigt, der indirekt über GPS realisiert wird. Dieser Ansatz ist wegen der eingangs beschriebenen Praxisuntauglichkeit nicht realisierbar und stünde nebenbei auch theoretisch dem Patent [51] entgegen, das GPS-Synchronisation in verteilten Systemen sehr allgemein abdeckt.

So haben sich mangels leicht anwendbarer proaktiver Synchronisationsmechanismen bei multimedialen Anwendungen reaktive Korrekturmaßnahmen etabliert. Wir stellen im Folgenden die verbreitetsten Ansätze im Audio-Bereich vor, die zumeist auch für andere Mediadaten (wie Video) anwendbar sind.

Machen wir uns dazu zu Beginn das Problem anhand eines einfachen verteilten Systems klar. In diesem, in der bereits aus Abschnitt 3.3 bekannten Abbildung 6.14, skizzierten Aufbau eines Erzeuger-Verbraucher Systems versendet der Sender einen Audiodatenstrom zum Empfänger, der diesen ausgibt – also eine typische Audio-Streaming Anwendung.

Beide Rechner verwenden die gleiche Abtastfrequenz f_{samp}, mit denen die Audiodaten bei A aufgezeich-

net und paketisiert werden. Nach der Übermittlung puffert B die Audiopakete zunächst in einem Abspielpuffer zur Kompensation von Übertragungs- und Verarbeitungsvarianzen und spielt die Daten zuletzt verzögert ab. Die Dimensionierung des verwendeten Abspielpuffers bestimmt den maximal kompensierbaren Jitter und die Systemlatenz, indem variable Paketzwischenankunftszeiten durch eine konstante Verzögerung ausgeglichen werden. Dass sich in dieser Beschreibung nahezu jede Audiodatenübertragung im Internet (sei es Streaming, IP-Telefonie oder auch NMP) widerspiegelt, resultiert daraus, dass dieses Prinzip Basis für jede paketbasierte Übertragung von kontinuierlichen Datenströmen ist.

Ein solches System läuft dann stabil, wenn die Abtastfrequenz f_{s_P} beim Produzenten gleich der beim Konsumenten ist, also $f_{s_C} = f_{s_P}$ gilt. Aus der einleitenden Betrachtung wissen wir, dass dieser Idealzustand in der Praxis nicht erreicht werden kann. Die immanente Ungenauigkeit der verwendeten Taktgeber wird in verteilten Systemen darüber hinaus durch eine erhöhte Wahrscheinlichkeit dafür verstärkt, bei den Endgeräten auf verschiedene technische Umsetzungen und unterschiedliche Umgebungsgrößen (bspw. Temperatur) zu treffen, die die Ungenauigkeiten verstärken. Bei der quantitativen Betrachtung müssen wir daher mit relativen Ungenauigkeiten $G_{f_{CP}} = 1 - \frac{f_{s_C}}{f_{s_P}}$ im Bereich zwischen 100 ppm und mehreren Promille rechnen, es gilt $|G_{f_{CP}}| = [10^{-4} \ldots 10^{-2}]$.

Der Effekt unterschiedlicher Taktraten beim Produzenten und Konsumenten ist wohlbekannt: erzeugt A pro Zeiteinheit mehr Daten, als B verarbeiten kann ($G_{f_{CP}} > 0$), kommt es bei B zu einem Pufferüberlauf, im umgekehrten Fall zu einem Unterlauf. Einen Eindruck davon, wann ein solcher Über- bzw. Unterlauf auftritt, können wir anhand eines typischen Aufbaus eines IP-Telefonie Systems gewinnen. Die Länge des Abspielpuffers nehmen wir hierbei mit 100 ms an, das initial mit 50 % gefüllt wird, bevor mit der Audioausgabe begonnen wird. Wir idealisieren das Netz und die Verarbeitung mit einem angenommen Jitter von Null, so dass bei $f_{s_C} = f_{s_P}$ der Abspielpuffer jederzeit zur Hälfte gefüllt ist. Liegt die Samplerate des Senders um 1 ‰ über der des Empfängers, läuft der Abspielpuffer nach 50 Sekunden über.

Ein Pufferüber- oder -unterlauf führt unmittelbar dazu, dass die weitere Audioausgabe fehlerhaft ist, da entweder Daten verloren gehen oder Lücken im Audiodatenstrom entstehen – für eine zuverlässige Funktion muss ein solches Ereignis daher proaktiv vermieden werden. Die angenommenen Parameter stellen typische Werte dar und erfordern daher von jeder Anwendung, die kontinuierliche (Audio)Daten in verteilten Systemen verarbeitet, geeignete Maßnahmen zur Kompensation unterschiedlicher Erzeuger- und Verbraucherfrequenzen.

Diese Aufgabe lässt sich grob in zwei Funktionseinheiten gliedern: im ersten Schritt muss ein Detektor erkennen, dass eine systematische Abweichung der verwendeten Frequenzen vorliegt, um dann im zweiten Schritt Korrekturmechanismen anwenden zu können.

Die Detektion von Asynchronitäten ist unter idealisierten Annahmen trivial: so wie ein initialer Füllstand des Abspielpuffers von 50 % gewählt wird, können Ober- und Untergrenzen definiert werden, die einen nahenden Überlauf bspw. bei 85 % bzw. einen Unterlauf bei 15 % ankündigen. Praxistauglich sind solche statischen Grenzen meist nicht, denn der Füllstand des Abspielpuffers wird wesentlich vom Jitter bestimmt, für dessen Kompensation er eigentlich vorgesehen ist. Eine kurze Stauung im Netz oder ein Burst von Paketen können den Puffer in den kritischen Bereich bringen und die Erkennung einer systematischen Frequenzunstimmigkeit verhindern. Die Erkennung und Zuordnung dieser überlagerten Einflüsse auf den Pufferzustand gelingt nur über eine kontinuierliche Beobachtung und Interpretation der Einzelmessungen, wie sie bspw. in [29] beschrieben ist.

Am Thema Detektion wird heute viel geforscht, ebenso wie an der Fragestellung der richtigen Dimensionierung von Abspielpuffern. In Zeiten, wo nahezu jedes mobile Gerät über Streaming-Funktionalität verfügt, in denen die RAM-Größe einen bedeutenden Kostenfaktor darstellt, wird die Entwicklung auch maßgeblich von kommerzieller Seite voran getrieben. Da diese prinzipbedingt immer eine Abwägung zwischen Latenz

und Fehlerwahrscheinlichkeit (im Sinne von Über- oder Unterlauf) ist, und letztere wiederum von den Eigenschaften im Netz bestimmt wird, bleibt das Thema weiterhin aktuell. Wir vertiefen es hier nicht weiter und werden im nächsten Abschnitt vorstellen, wie wir im NMP unsere Detektion indirekt aus bereits vorhandenen Funktionseinheiten gewinnen.

Blicken wir stattdessen weiter auf die zweite Funktionseinheit, der proaktiven Korrektur. Wir nehmen dazu weiter an, der Detektor hat eine systematische Abweichung der Abtastraten zwischen Sender und Empfänger erkannt. Wir bleiben bei unserem Beispiel der IP-Telefonie und nehmen an, dass eine Gangabweichung von $G_{f_{CP}} = 10^{-3}$ erkannt wurde, was Abtastraten von 48 kHz beim Empfänger bzw. 48.048 kHz beim Sender entspricht. So empfängt B im Mittel während der Zeiteinheit, in der er 1000 Samples abspielt, 1001 Samples von A.

Da die Abtastraten der Audiohardware beim Sender und Empfänger nicht direkt justierbar sind, bleibt dem Empfänger nur eine Möglichkeit, einen Überlauf seines Abspielpuffers zu verhindern – die empfangenen Daten müssen in einem höheren Tempo konsumiert, als sie von der Audiohardware verarbeitet werden. Praktisch anwendbar ist dabei nur das Verwerfen von Daten, theoretisch besteht darüber hinaus die Möglichkeit einer Abtastratenanpassung.

Bei der zweit genannten Methode müssten in diesem Fall 1001 empfangene Samples zeitlich gestaucht und unter Beibehaltung des Signalverlaufs auf 1000 Abtastwerte herunter gerechnet werden. Der dafür erforderliche Rechenaufwand steht in keinem Verhältnis zum Qualitätsgewinn dieser Methode gegenüber dem einfachen Verwerfen von Abtastwerten. Laut [69] bemerken Anwender das Verwerfen von Daten bis hinauf zu 0.25 Prozent aller Samples nicht, was in diesem Fall eine Re-Synchronisation ermöglicht. Mit geeigneten Maßnahmen, wie der Auswahl der zu verwerfenden Samples unter minimaler Verfälschung des Audiosignals (bspw. Verwurf eines mit identischen Nachbarwerten), kann diese Korrektur so ausgeführt werden, dass sie praktisch auch bei einem Verwurf von bis zu einem Prozent nicht wahrnehmbar ist.

Ist im umgekehrten Fall die Verarbeitungsrate beim Empfänger höher als beim Sender, muss ein Puffer-Unterlauf verhindert werden, indem der Empfänger die Datenmenge vor der Audioausgabe erhöht. Analog zum Verwerfen von Samples werden Abtastwerte verdoppelt, um die Netz-Datenrate mit der Samplerate der Audiohardware abzugleichen. Die Suche nach geeigneten Positionen, wo im Datenstrom Samples verdoppelt werden können, bleibt im Prinzip gleich. Auch hierbei empfiehlt es sich, Manipulationen an Orten lokaler Konstanz vorzunehmen, also an Signalabschnitten mit geringer Steigung.

Im Weiteren wollen wir zwischen Pufferüber- und Unterläufen im Interesse einer besseren Lesbarkeit nicht differenzieren, denn des einen Unterlauf ist des anderen Überlauf und führt in jedem Fall zu Diskontinuitäten im Datenstrom, die sich auf die auditive Wahrnehmung gleichermaßen nachteilig auswirken. Wenn im Folgenden von Pufferüberläufen die Rede ist, dann sind – wenn nicht anders erwähnt – implizit auch Unterläufe eingeschlossen.

6.7.3 NMP spezifische Besonderheiten bei der Synchronisation

So prinzipiell vergleichbar die Abläufe in einer IP-Telefonie Anwendung und bei NMP auf den ersten Blick sind – bei genauer Betrachtung erweist sich die Anwendbarkeit existierender Verfahren aufgrund der besonderen Anforderungen für die Praxis als nicht gegeben. Rufen wir uns dazu noch einmal die für die Bearbeitung im Server verfügbaren Zeiten und die daraus abgeleiteten Puffergrößen in Erinnerung. Als grobe Vorgabe wollen wir eine maximale Verweilzeit der Audiodaten im Server von 10 ms nicht überschreiten. Bei Verwendung unserer Standard Paketgröße von 2.66 ms hat der Synchronisationspuffer im Server daher maximal Platz für vier Pakete.

Idealerweise sollte der Synchronisationspuffer zu 50 % gefüllt sein und ist dann dabei jeweils zwei Pa-

kete von einem Unter- bzw. Überlauf entfernt. Laut Latenzanalyse in Kapitel 5 müssen wir von dieser Reserve ein Paket für die Gleichtaktung der asynchron zueinander produzierten und eintreffenden Audiopakete vorsehen – auch unter idealen Bedingungen muss daher im Server ein Paket von jedem Client vorgehalten werden, bis das dazugehörige Paket vom letzten Client empfangen ist und das Mischen begonnen werden kann. Somit verbleibt eine Reserve von weniger als zwei Paketen, innerhalb derer eine zuverlässige Detektion unterschiedlicher Datenraten erfolgen muss und geeignete Kompensationsmaßnahmen anzuwenden sind.

Auf den Entwurf des Detektionsmechanismus wirkt sich bei NMP insbesondere die systemimmanente Existenz von Paketverlusten erschwerend aus: gerade weil unser Ansatz darauf abzielt, eine Anwendung mit minimaler Latenz bei tolerierbaren Paketverlusten zu schaffen, dürfen Überläufe das gewählte Verfahren nicht beeinträchtigen. Mit unserer anvisierten maximal tolerierbaren Paketverlustrate von bis zu 1 % und einer Paketrate von 375 Hz bedeutet das, dass pro Sekunde mit drei Paketverlusten zu rechnen ist, die sich als Überlagerung von unterschiedlichen Sampleraten der Clients untereinander und Netz- und Verarbeitungsjitter in den Endsystemen ergeben.

NMP spezifische Einschränkungen gibt es auch bei der anschließenden Ausgestaltung der Kompensationsmechanismen, sobald eine systematische Abweichung zwischen den Sampleraten der Clients untereinander erkannt wurde. Die einleitend vorgestellte Methode der Abtastratenanpassung durch Entfernen oder Hinzufügen einzelner Samples ist aus mehreren Gründen nicht anwendbar. Zum einen erfordert die paketbasierte Datenverarbeitung in der Audiohardware ein Aufbrechen der Paketgrenzen und damit einhergehend eine zusätzliche Pufferverzögerung. Entfernt in einer IP-Telefonie Anwendung Rechner B ein Sample aus dem Paket p_n mit der Sequenznummer n, dann muss er auf Paket p_{n+1} warten, um das erste Sample von p_{n+1} an p_n anzuhängen und der Soundkarte wieder einen vollständigen Audioblock übergeben zu können. Das Entfernen oder Hinzufügen einzelner Samples führt so zu einer Erhöhung der Systemlatenz um t_φ und erfordert eine Re-Paketisierung, die so lange anhält, bis sich aufeinanderfolgende Modifikationen zu einer vollen Blockgröße aufsummiert haben.

Ein solches Verfahren, das die Systemlatenz signifikant erhöht, eignet sich für NMP grundsätzlich nicht. Aber auch aus praktischen Erwägungen scheint die Modifikation des Audiodatenstroms um einzelne Abtastwerte wenig sinnvoll. Nehmen wir an, wir haben in einer laufenden NMP Sitzung erkannt, dass Client B eine um 1 ‰ höhere Abtastrate verwendet als Client C und entschließen uns – trotz der inakzeptablen Auswirkung auf die Latenz – im Audiodatenstrom von B jedes 1000ste Sample zu verwerfen. Bei einer Paketgröße von 128 Samples wird so etwa in jedem achten Paket ein Sample gelöscht, die Daten werden anschließend zu neuen Paketen von 128 Samples zusammengefasst. Nach im Mittel 100 Paketen muss jedoch mit einem System immanenten Paketverlust gerechnet werden, der die ursprüngliche Planung zunichtemacht. Betrifft das Ereignis Client B, hätte man an dieser Stelle den Audiodatenstrom von B gleich um 128 Samples kürzen können, indem die Lücke unmittelbar durch das folgende Paket geschlossen wird. Gleiches gilt für den Fall, dass der Überlauf bei Client C lag: es wird zu einer Diskontinuität kommen, die dazu genutzt werden kann, den Audiodatenstrom an dieser Fehlstelle zu dehnen und so die Kompensation von Asynchronität als Teil immanenter Paketverluste zu behandeln.

Somit sind gängige Methoden für Detektion und Korrektur von Asynchronitäten in NMP nicht verwendbar. Im folgenden Abschnitt leiten wir unseren Entwurf für eine Synchronisation her, der mit den diskutierten Einschränkungen umgehen kann und darüber hinaus in der Lage ist, Asynchronitäten über die ohnehin vorhandenen Mechanismen zur Korrektur von Paketverlusten zu kompensieren.

6.7.4 Entwurf einer Zeitreferenz losen Synchronisation

Wir beginnen unseren Entwurf mit einer Anforderungsanalyse, aus der wir die erforderlichen Eigenschaften einer geeigneten Synchronisationsmethode für NMP definieren. Das so entworfene Verfahren wird anschließend diskutiert und bewertet.

6.7.4.1 Anforderungsanalyse

Bei der bisher beschriebenen Synchronisationsproblematik wird die prinzipielle Frequenzanpassung Empfänger-seitig so durchgeführt, dass der Datenstrom gestaucht wird, wenn die Verbrauchsrate geringer als die Erzeugerrate ist und im umgekehrten Fall gestreckt wird. Wir haben dabei angenommen, dass der Empfänger für die Korrektur verantwortlich ist, weil diesem Erzeuger- und Verbraucherfrequenz bekannt sind. Es spricht andererseits nichts dagegen, die Gangabweichung dem Sender explizit bekannt zu machen und die Synchronisierung dort vornehmen zu lassen. In beiden Fällen werden dem verwendeten Verfahren eine Referenz- und eine oder mehrere zu korrigierende Raten vorgegeben.

In einem NMP-System bietet sich intuitiv der Server als Referenz für die Synchronisation an, insbesondere weil diese Funktion als eine seiner Kernaufgaben definiert wurde. Bei genauerer Betrachtung wird allerdings klar, dass dieser von allen beteiligten Endsystemen die ungünstigste Wahl wäre. Das veranschaulichen wir kurz an einem beispielhaften Aufbau eines NMP-Systems mit einem Server S und zwei Clients B und C.

Bei Server S treffen von den Clients B und C idealerweise alle $t_\varphi = \frac{8}{3}\,ms$ je ein Paket ein. Während die Paketrate über die Samplerate der Audiohardware und der gewählten Blockgröße definiert wird, kann der Server sein Zeitsignal aus unterschiedlichen Quellen gewinnen. Das kann die Echtzeituhr (RTC, Real Time Clock), der Takt dedizierter Peripheriebausteine (wie Interrupt Controller) oder auch der Prozessortakt sein. Die Genauigkeit der damit berechneten Rechneruhr variiert von verwendeter Quelle und Betriebssystem unterschiedlich stark. Wird der Server zusätzlich an ein Zeitnormal, bspw. über NTP, synchronisiert, können sich Unstetigkeiten an den Synchronisationspunkten ergeben. Die Rechneruhr wäre daher eine relativ unzuverlässige Referenz für die Synchronisation.

Darüber hinaus lässt sich sehr anschaulich zeigen, dass es keinen Sinn macht, die Clients an eine vom Server vorgegebene Frequenz zu synchronisieren. Dazu nehmen wir einen Server mit idealer Rechneruhr an, während die Clients B und C mit einer von der Nennfrequenz deutlich abweichenden Abtastrate von 49 kHz betrieben werden. Wäre die Rechneruhr des Servers die Referenz für die Taktanpassung, müssten dort $\frac{1}{49}$ aller Audiodaten von beiden Clients verworfen werden, um die Abtastraten an die Rechneruhr anzupassen. Anschließend müssten die Clients den zurückgesendeten Datenstrom des Mischsignals wieder auf die Abtastrate der Audiohardware strecken, um die erforderliche Datenmenge ausgeben zu können. Im Ergebnis wird durch diese Datenratenanpassung bei beiden Clients jeweils eine Fehlerrate von $\frac{2}{49}$ verursacht. Die unveränderten Datenströme hätten dagegen zwar eine absolut höhere Abtastrate, blieben aber synchron.

Diese beispielhaft idealisierte Annahme wird man in der Praxis nicht antreffen, d.h., die Abtastraten der Clients untereinander werden sich immer unterscheiden. Aber auch dann ist es offensichtlich sinnvoller, eine der verwendeten Abtastraten als Referenz zu verwenden, weil damit die Anzahl erforderlicher Manipulationen der Datenströme minimiert wird. Die Wahl der Referenz spielt dabei dann keine Rolle, wenn Verwerfen und Hinzufügen von Audiodaten als auf die wahrgenommene Audioqualität im gleichen Maße schädigend betrachtet werden. Wählt man als Referenz den Client mit der höchsten Abtastrate, müssen bei allen anderen Clients Audiodaten hinzugefügt werden, fällt die Wahl auf den langsamsten, müssen bei den übrigen Daten verworfen werden.

Aus diesen Überlegungen ergeben sich an das in NMP zu verwendeten Synchronisationsverfahren folgende Anforderungen:

Einhaltung der Paketgrenzen

Um keine zusätzliche Verzögerung für eine Neukomposition von Audiopaketen zu verursachen, dürfen Datenratenanpassungen nur auf Paketebene erfolgen, d.h., es sind Pakete entweder vollständig zu verwerfen oder hinzuzufügen.

Die Audiohardware bestimmt den Referenztakt

Obwohl als Synchronisationskomponente vorgesehen, stellt der Server nicht den Referenztakt. Um die Anzahl der Datenanpassungen zu minimieren, muss die Samplefrequenz eines Clients als Referenz gewählt werden.

Robustheit gegen Paketverluste

Die Erkennung und Korrektur von Drifts zwischen den Sampleraten muss immanent vorhandene Paketverluste tolerieren können.

6.7.5 Reaktive Synchronisation

Das erste von uns in NMP verwendete Synchronisationsverfahren basiert auf statisch dimensionierte Warteschlangen und ist mit einigen Erweiterungen in der Lage, die vorgestellten spezifischen Probleme zu berücksichtigen. Folgend erläutern wir das Funktionsprinzip dieses Verfahrens und untersuchen sein Verhalten hinsichtlich der genannten Anforderungen.

Das gewählte Verfahren beruht im Prinzip auf ein De-Jitter Puffer Array für jeden NMP-Client in Form einer Paket Warteschlange (*Queue*). Auf die Wahl geeigneter Datenstrukturen für die Umsetzung unseres De-Jitter Puffers wird in Kapitel 7 näher eingegangen. Vorgreifend darauf sind für ein Verständnis der folgenden Diskussion die verfügbaren Operationen der Warteschlangen zu erläutern. Die für das Mischen benötigten Audiopakete werden immer vom Kopf der Queue entnommen, wohingegen das Einfügen empfangener Pakete einen wahlfreien Zugriff erfordert, um Diskontinuitäten in der Monotonie der Sequenz durch Störungen bei der Übertragung im Netz tolerieren zu können. Wenn wir im Folgenden von Warteschlangen und Queues sprechen, beziehen wir uns auf eine um den wahlfreien schreibenden Zugriff erweiterte Form der im Allgemeinen bekannten Art, die nur einen Zugriff auf das erste (*dequeue*) und letzte Element (*enqueue*) erlauben. Auch werden wir die dort verwendeten Termini verwenden, bspw. wird mit Kopfelement der vorderste Eintrag in der Queue bezeichnet.

Die zu verarbeitenden Audiopakete werden vom einem Client c mit inkrementell streng monoton steigenden Sequenznummern versehen und nach dem Empfang beim Server in die Queue q_c eingefügt. Bei Erreichen eines definierten Zustands werden die Kopfelemente entnommen und dem Mischer zur Verarbeitung zugeführt. Wie aus unseren eingangs erläuterten Erwägungen sind für eine maximale Verweilzeit der Daten im Server von knapp 10 ms höchstens vier Audiopakete vorzuhalten ($4 \cdot t_\varphi = 10.66\,ms$).

q_A, q_B, q_C	Puffer Queues für Client A, B, C
$seq_{h(A)}$, $seq_{h(B)}$, $seq_{h(C)}$	Sequenznummern der Kopfelemente in q_A, q_B und q_C
$seq_{t(A)}$, $seq_{t(B)}$, $seq_{t(C)}$	höchste bisherigen Sequenznummern in q_A, q_B und q_C
$num_{p(A)}$, $num_{p(B)}$, $num_{p(C)}$	Anzahl enthaltener Pakete in q_A, q_B und q_C
$[k,l,m]$	aus den Paketen $p_{A(k)}$, $p_{B(l)}$ und $p_{C(m)}$ gemischtes Paket
$seq(p)$	Sequenznummer des Paketes p
$pos(p) = seq(p) - seq_h$	Positionsindex für Paket p
$\vec{s_v}$	Versatzvektor zwischen den Sequenzen in einem gemischten Paket
$\emptyset(n)$	Leeres Paket mit Sequenznummer n ohne Audiodaten

Tabelle 6.7: Verwendete Bezeichner

Die Queues sind in diesem Beispiel als q_A, q_B und q_C bezeichnet, wobei das zu mischende Pakete immer vom Kopf entnommen wird und die Sequenznummer $seq_{h(A/B/C)}$ hat. Die Position eines empfangene Paketes p muss nicht notwendigerweise das erste freie Element der Warteschlange sein, da Paketverluste, Umordnungen oder Verdoppelungen auftreten können. Sie ist vielmehr aus der Differenz seiner Sequenznummer und der des Kopfelements $pos(p) = seq(p) - seq_h$ zu ermitteln. Ist an der errechneten Position bereits ein Paket enthalten, hat man es mit einer Paketverdopplung zu tun, die ignoriert werden kann. Verspätete und für das Mischen nicht mehr verwendbare Pakete haben eine negative Position und können ebenso ignoriert werden. Eine Differenz größer oder gleich der Queue-Länge (hier $pos(p) \geq 4$) signalisiert einen Pufferüberlauf, da mehr Pakete empfangen als verarbeitet wurden.

Eine für die Betrachtung weitere wichtige Kenngröße ist die Anzahl der in einer Queue vorgehaltenen Elemente num_p. Im Idealfall (d.h., ohne die genannten potentiellen Störungen bei der Datenübertragung) entspricht diese der um eins erhöhten Position des zuletzt eingefügten Paketes, $num_p = pos_p + 1$. Für die korrekte Funktionsweise auch unter realen Bedingungen müssen wir einen dedizierten Wert für das Paket mit der bisher höchsten eingereihten Sequenznummer seq_t mitführen. Dieser weicht nur bei Diskontinuitäten von dem – in der überwiegende Zeit gültigen – Wert für num_p ab, dann gilt $num_p = seq_t - seq_h + 1$.

Bis hierher entspricht die Darstellung allgemeinen Synchronisationsverfahren in typischen Medienanwendungen. Eine Besonderheit des NMP-Servers ist jedoch die oben beschriebene externe Taktung durch die Clients. Der Server kann die Bedingungen, die das Mischen auslösen, nicht wie sonst üblich über Zeitmessungen gewinnen, sondern muss diese Information aus den Sequenznummern und Füllständen der Queues ableiten können. Das sollte während einer laufenden NMP Sitzung durchaus möglich sein, denn in diesem Zustand treffen die Pakete quasi-isochron beim Server ein, d.h., die Einzelwerte der Paketzwischenankunftszeiten werden zwar um den Idealwert t_φ schwanken, im Mittel dabei jedoch soweit im Rahmen bleiben, dass die Synchronisation einer Sitzung möglich ist.

Diese funktionale Voraussetzung verwenden wir für die Festlegung von Kriterien, die eine Bestimmung der Mischzeitpunkte ohne Zeitreferenz allein anhand der Zustände des Queue Arrays ermöglichen. Die Anzahl erforderlicher Kriterien ist überraschend klein. Wir listen diese zunächst folgend auf und weisen anschließend nach, dass damit alle zu berücksichtigenden Situationen abgedeckt werden.

Im vereinfachten Algorithmus 6 ist dargestellt, wie für die vorgestellte Dimensionierung des Warteschlangen Arrays nach jedem Empfang eines Audiopaketes anhand folgender Richtlinien ermittelt wird, ob ein Mischvorgang möglich oder nötig ist:

R1/PULL: vollständiges Mischen so früh wie möglich

$R1 = ((num_{p(A)} > 0) \wedge (num_{p(B)} > 0) \wedge (num_{p(C)} > 0))$

Mische wenn von allen Clients mindestens ein Audiopaket vorhanden ist.

R2/PUSH: partielles Mischen so spät wie nötig

$R2 = (num_{p(A)} > 4) \vee (num_{p(B)} > 4) \vee (num_{p(C)} > 4))$

Erzwinge Mischen frühestens, wenn eine Queue überläuft.

Bereits diese beiden Vorgaben reichen aus, um den Widrigkeiten von Netzwerk- und Anwendungsjitter, Paketverlusten und Frequenzdrift zwischen den Clients zu begegnen. Graduelle Verbesserungen sind mit ergänzenden Maßnahmen unseres einfachen Basisverfahrens möglich. Auf diese werden wir im Interesse einer besseren Verständlichkeit später kommen, nachdem wir die Funktionsfähigkeit und Robustheit unseres Verfahrens anhand einiger ausgewählter Szenarien nachgewiesen haben.

Algorithmus 6 : Basis-Synchronisation bei einheitlicher Länge der Synchronisationspuffer

Input : Aufruf nach Eingang eines Audiopaketes
Input : `isR1()` wahr, wenn Kopfelement in allen Queues vorhanden
Input : `isR2()` wahr, wenn mindestens eine Queue voll
if *(isR1() || isR2())* **then**
    `mix_pakete[] = dequeue(alle nicht-leeren Queues)`
    `mische(mix_pakete[])`
end if

6.7.5.1 Erkennung und Bearbeitung von Paketverlusten

Bei der einleitenden Diskussion NMP-spezifischer Eigenschaften der Synchronisation haben wir festgestellt, dass ein signifikant hoher Anteil der zu behandelnden Unregelmäßigkeiten Paketverluste auf dem Übertragungsweg von den Clients zum Server sind. In diesem Absatz wollen wir eine Festlegung treffen, anhand welcher Kriterien Paketverluste als solche detektiert werden und wie damit umgegangen wird.

Der Vollständigkeit halber sei eingangs darauf hingewiesen, dass in unserem NMP-System eine Sequenzierung der Audiopakete seitens der Clients erfolgt. Diese Sequenznummer wird auf dem Weg eines Paketes nicht verändert, d.h., ein Client kann anhand dieser ein vom Server reflektiertes Mischpaket eindeutig einem versendeten Paket zuordnen. Gleichzeitig kann ein Server anhand dieser Paketverluste erkennen, die sich dann naheliegend und trivial an einer Abweichung der erwarteten von der empfangenen Sequenznummer äußern.

Hat der Server so bspw. von einem Client A als letztes Paket $p_{A(k-1)}$ erhalten, dann erwartet er als nächstes den Empfang von $p_{A(k)}$. Wird stattdessen ein Paket mit einer abweichenden Sequenznummer $seq(p)$ empfangen, handelt es sich um einen Fehler, der sich in zwei unterschiedlichen Ausprägungen niederschlägt.

$seq(p) > k$

: Eine höhere Sequenznummer als die erwartete kann entweder nach einem verworfenen Paket oder einer Umordnung der Pakete bei der Übertragung empfangen werden. Da Quell- und Zielrechner während einer Sitzung statisch sind, kann eine Umordnung nur dann eintreten, wenn sich der Netzpfad der Übertragung ändert. Ein solches Ereignis hätte auf eine laufende Sitzung entweder einen Abbruch zur Folge oder würde aufgrund geänderter Paketlaufzeiten eine einmalige Stauung verursachen. Diese wird, wie wir in einem der nächsten Abschnitte genauer untersuchen werden, gesondert erkannt, so dass wir in einer solchen Situation immer von Paketverlusten ausgehen können.

$seq(p) < k$

: Eine niedrigere Sequenznummer als die erwartete kann das Ergebnis einer Umordnung oder einer Paketverdoppelung sein. Diese können leicht danach unterschieden werden, ob ein Paket mit entsprechender Sequenznummer von diesem Client bereits in der Puffer-Queue enthalten ist. In diesem Fall kann das doppelt empfangene Paket verworfen werden, andernfalls ist es wie ein reguläres Paket zu bearbeiten.

Diese potentiell zu berücksichtigenden Fehlerquellen haben sich während unserer Arbeit als rein theoretischer Natur erwiesen: in keiner unserer Untersuchungen und Sitzungen wurden andere Fehlerursachen als verworfene Pakete detektiert. Eine Abweichung in der Sequenznummer bedeutet daher praktisch immer, dass alle Pakete zwischen der erwarteten und der empfangenen Sequenznummer als Verluste gewertet werden können. Da wir ohne Neuanforderung verlorener Pakete arbeiten, kann diese Lücke vom Synchronisationsverfahren sofort bearbeitet werden.

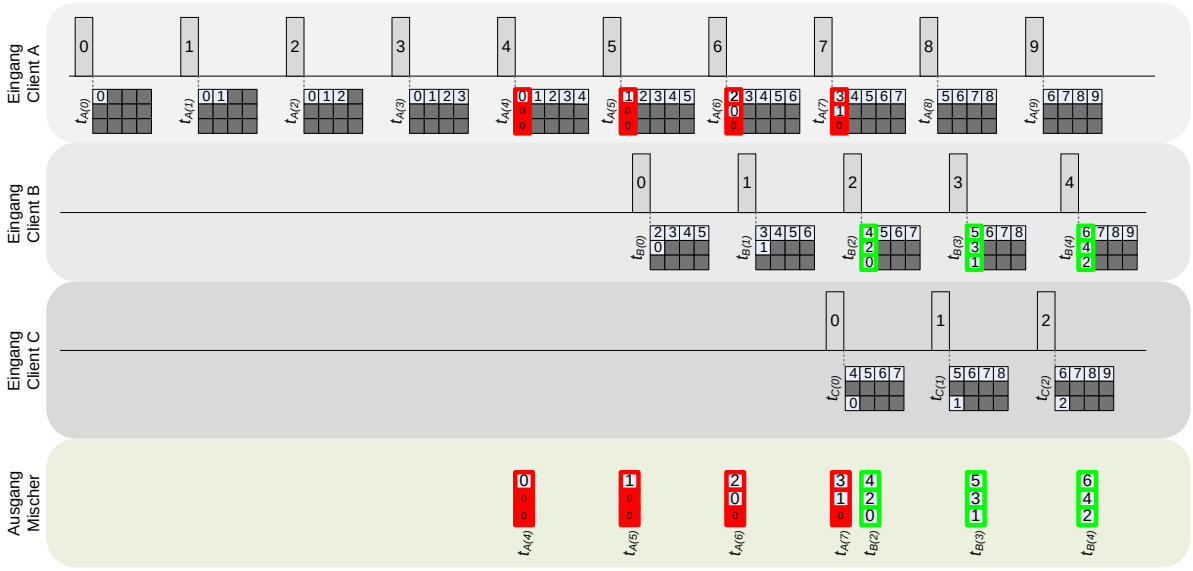

Abbildung 6.15: Zustand der Puffer Queues während der Aufbauphase

6.7.5.2 Aufbau einer NMP Sitzung: von der Aufbau- zur stabilen Phase

In unserer Implementation unterscheiden wir bei NMP Sitzungen zwei Phasen: in der so genannten Aufbauphase findet die Initiierung der Sitzung statt, innerhalb derer Teilnehmer nach und nach zur bereits laufende Sitzung beitreten oder diese wieder verlassen. Dagegen wird beim Musizieren in der anschließenden stabilen Phase eine statische Zusammensetzung angenommen.

Auch ohne diese explizite Unterteilung gibt es eine einleitende Phase, in der die Audiodaten der Clients nach und nach den Server erreichen. Erst wenn sich die Teilnehmerkomposition nicht mehr ändert und sich der kontinuierliche Empfang von Audiopaketen aller Clients einstellt, ist der stabile Zustand einer Sitzung erreicht. Ein solcher Aufbau einer Sitzung ist in Abbildung 6.15 dargestellt.

In diesem Aufbau beginnen die Clients A, B und C zeitlich versetzt mit dem Versenden der Audiopakete. Vom Client A treffen die Daten als erstes am Server ein und werden in q_a eingefügt. Bei den Paketen mit den Sequenznummern 0 bis 3 ($p_{A(0)}$, $p_{A(1)}$, $p_{A(2)}$ und $p_{A(3)}$) geschieht dies, ohne dass ein Mischvorgang eingeleitet wird. Nach Eingang von Paket Nummer 4 kommt Richtlinie R2 zum tragen: der Puffer für Client A ist zum Zeitpunkt $t_{A(4)}$ voll. Um das zuletzt erhaltene Paket $p_{A(4)}$ einreihen zu können, muss ein Mischvorgang ausgeführt und ein Paket verbraucht werden. Dieses erzwungene Mischen aufgrund der Richtlinie R2 ist in der Abbildung so gekennzeichnet, dass das partiell gemischte Paket rot markiert ist. Zu diesem Zeitpunkt sind die Queues von B und C noch leer, das gemischte Paket beinhaltet also nur die Daten von Client A. Gleiches geschieht zum Zeitpunkt $t_{A(5)}$, an dem nur das Audiopaket $p_{A(1)}$ für den Mischer verfügbar ist.

Kurz darauf trifft zum Zeitpunkt $t_{B(0)}$ das erste Paket von B beim Server ein. Nach dem Einfügen in die Warteschlange sind die Pakete $p_{A(2)}$ und $p_{B(0)}$ verfügbar. Solange die Queue von C jedoch leer ist, kommt Richtlinie R1 nicht zum tragen. Stattdessen wird wieder eine Aktion aufgrund von R2 erzwungen, sobald das nächste Paket von A empfangen wird. Zum Zeitpunkt $t_{A(6)}$ werden $p_{A(2)}$ und $p_{B(0)}$ partiell gemischt und machen in der Warteschlange Platz für das Einfügen von $p_{A(6)}$. An diesem Zustand ändert auch die nächste Runde nichts: nach dem Eingang von $p_{B(1)}$ und $p_{A(7)}$ läuft Puffer A erneut über und erzwingt das Mischen von $p_{A(3)}$ und $p_{B(1)}$.

Erst kurz danach trifft zum Zeitpunkt $t_{C(0)}$ das erste Audiopaket von Client C beim Server ein. Der Zustand der Warteschlangen erlaubt noch keine Anwendung der Richtlinien: Puffer A ist voll aber noch nicht

übergelaufen. Gleichzeitig liegt jetzt zwar ein Audiopaket von C vor, von Client B wurden dagegen alle bisher empfangenen Pakete bereits verarbeitet, R1 ist somit noch nicht anwendbar. Das ändert sich mit dem Empfang von $p_{B(2)}$: alle Queues haben ein Kopfelement und genügen somit R1. Der Mix der Pakete $p_{A(4)}$, $p_{B(2)}$ und $p_{C(0)}$ beinhaltet erstmalig Daten aller teilnehmenden Clients und markiert damit den Übergang von der Initialisierung in den stabilen Zustand. Zur grafischen Unterscheidung sind diese durch R1 ausgelösten Mischvorgänge grün gekennzeichnet.

Die stabile Phase ist in der Abbildung als eine sich wiederholende Abfolge von Zuständen erkennbar, an deren Ende die Richtlinie R1 ein Mischen der von allen Clients verfügbaren Audiopakete steht. Der Füllstand von q_a ($num_{p(A)}$) pendelt dabei zwischen drei und vier, es kommt jedoch nicht mehr zu Überläufen und somit keiner weiteren Ausführung von R2. Die Warteschlange q_c ist entweder leer oder enthält so lange ein Element, bis das folgende Paket von B eintrifft. Dann sind wieder von allen Clients Daten für einen Mischvorgang über R1 vorhanden. Da dieses unmittelbar nach dem Empfang des Paketes von B erfolgt, wird die Queue q_b sofort wieder geleert und enthält praktisch dauerhaft keine Elemente.

Dieser beschriebene Ablauf und das daraus resultierende Verhalten sind offensichtlich nur unter der Annahme eines idealen Systems gegeben, in dem die Paketzwischenankunftszeiten konstant bei t_φ liegen und somit kein Jitter vorhanden ist, sowie Paketverluste und Asynchronitäten aufgrund abweichender Abtastraten nicht auftreten. Wie diese das gewählte Verfahren beeinflussen, analysieren wir anschließend in eigenen Abschnitten. Zunächst fassen wir die Merkmale und den Ausgang der Initiierungsphase zusammen.

Per Definition ist das zentrale Merkmal der Anlaufphase das Fehlen des Datenstroms von mindestens einem Client. Die in dieser Phase von den anderen Clients bereits empfangenen Pakete werden in die Warteschlangen eingereiht und lassen deren Füllstand langsam ansteigen. Läuft eine Queue voll, bevor von allen Clients Daten vorliegen, kommt es zu einem Überlauf und einem nach R2 erzwungenen partiellen Mischvorgang. Der Client, dessen Pakete am ehesten beim Server angekommen sind, bestimmt somit in dieser Phase die Mischzeitpunkte dadurch, dass der Füllstand seiner Queue am höchsten ist und am ehesten überläuft. Diese Bedingung bleibt auch beim sukzessiven Beitritt zusätzlicher Clients bestehen, da deren Audiodaten bereits beim nächsten Überlauf verbraucht werden. Der Übergang in die stabile Phase erfolgt mit dem ersten Eintreffen der Daten des zuletzt beigetretenen Clients. Ab dann ist aufgrund von Richtlinie R1 ein vollständiges Mischen möglich, unter idealen Bedingungen ist dies dauerhaft, da sich ein periodischer und kontinuierlicher Zustandswechsel alle t_φ einstellt.

Der auslösende Faktor für das Mischen ist im stabilen Zustand fortan nicht mehr der Überlauf einer Queue, sondern das Eintreffen des letzten zugehörigen Audiopaketes innerhalb eines Zyklus. In unserem diskutierten Beispiel wird der stabile Zustand mit dem Mischen des Paketes $[4,2,0]$ betreten. Der Versatz der Sequenzen zueinander bleibt in einem idealen System konstant, so dass im dargestellten Fall ein Paket $p_{A(n)}$ jeweils mit $p_{B(n-2)}$ und $p_{C(n-4)}$ gemischt wird. Diesen Wert werden wir später benötigen und führen dafür hier die Bezeichnung *skew vektor* $\vec{s_v}$ ein, der für ein gemischtes Paket den Versatz der Sequenznummern untereinander beinhaltet. Als Kenngröße werden wir diesen Wert jeweils so umwandeln, dass der kleinste Wert als Referenz verwendet wird:

$$\vec{s_v} = \begin{pmatrix} seq_a - seq_{min} \\ seq_b - seq_{min} \\ seq_c - seq_{min} \end{pmatrix} \qquad mit \quad seq_{min} = min\{seq_a, seq_b, seq_c\}$$

Idealerweise ist dieser Vektor in der stabilen Phase konstant, wie bspw. in unserem Beispiel $\vec{s_v} = (4,2,0)$.

Zu beachten ist, dass die hier als initial bezeichnete Phase nicht ausschließlich auf dem Aufbau der Sitzung beschränkt ist. Die Benennung resultiert daraus, dass das beschriebene Verhalten jedem Sitzungsaufbau innewohnt. Gleichwohl kann ein Übergang aus der stabilen in die initiale Phase erfolgen, wenn die

Verbindung zu einem Teilnehmer gestört und der Datenfluss länger unterbrochen wird. Dann erfolgt die Verarbeitung der Audiopakete über R2 initiierte partielle Mischvorgänge. Nach einer Behebung der Störung und dem erneuten regulären Empfang aller Daten wird anschließend wieder über R1 vollständig gemischt. Analog zur beschreibenden Benennung der betreffenden Richtlinien wählen wir im Folgenden ergänzend die Bezeichner PUSH- und PULL-Phase zur Unterscheidung dieser beiden Zustände.

Zwei Eigenschaften dieses Verfahrens erscheinen zu diesem Zeitpunkt fragwürdig: zunächst fällt auf, dass der Versatz in der stabilen Phase nicht dem zeitlichen beim Empfang entspricht. So erreichen den Server zu Beginn sechs Pakete von A, bevor das erste Paket von B eintrifft, trotzdem beträgt der endgültige Versatz nur zwei Pakete. Zur Klärung müssen wir uns hierfür vor Augen führen, dass wir eine relative Synchronisierung anstreben, d.h., der NMP-Server stellt lediglich sicher, dass beim Mischen ein gleich bleibender Versatz beibehalten und eine maximale Verweildauer garantiert wird - die absolute Synchronität wird von den Musikern adaptiv geregelt, so wie es auf der realen Bühne praktiziert wird.

Die zweite Auffälligkeit bei Erreichen des stabilen Zustands ist der Füllstand der Queues: q_a ist die meiste Zeit voll, q_b und q_c sind dagegen nahezu immer leer. Diese Eigenschaft widerspricht den Vorgaben für ein De-Jitter Puffer, in dem ein mittlerer Füllstand von 50 % anzustreben ist. Tatsächlich benötigen wir im NMP-Server kein De-Jitter Puffer im ursprünglichen Sinn, denn eine Isochronität bei der Bearbeitung der Audiopakete muss nicht erfüllt werden. Diese ist erst wieder beim Client erforderlich und wird dort über ein Abspielpuffer realisiert. Da der Server bestrebt ist, Daten so früh wie möglich zu verarbeiten und ein Pufferunterlauf prinzipbedingt nicht auftreten kann, enthalten die Queues im Idealzustand maximal ein Element. Dieser stellt sich in unserem Beispiel für q_b und q_c auch tatsächlich ein.

Augenscheinlich scheint es weiterhin sinnvoll zu sein, q_a durch Manipulation des Versatzes ebenfalls in diesen Idealzustand zu bringen und die volle Queue als Reserve für Jitter nutzen zu können. Wir werden im folgenden Abschnitt den Einfluss von Jitter und Bursts auf das Verfahren untersuchen und dabei diesen offenen Punkt klären.

6.7.5.3 Jitter und Bursts

Aus der Abbildung 6.15 geht unmittelbar hervor, dass im stabilen Zustand ab Zeitpunkt $t_{B(2)}$ kein De-Jitter Puffer nötig wäre, wenn wie dargestellt alle Clients konstante Paketzwischenankunftszeiten haben. Dann genügt es, von jedem Client nur das aktuelle Paket zwischenzuspeichern und auf die korrespondierenden Pakete der anderen Clients zu warten. In diesem idealen Fall reduziert sich die maximale Verweilzeit der Audiodaten im Server auf t_φ, so wie es in der Latenzanalyse in Kapitel 5 als untere Schranke definiert wurde.

In einem realen NMP Aufbau erwarten wir dagegen burstartige Paketzwischenankunftszeiten, die sich aufgrund von variierenden Verarbeitungszeiten in den Netzkomponenten und Endsystemen ergeben. In Abbildung 6.16 ist der stabile Zustand einer NMP Sitzung mit drei Clients dargestellt, der unterschiedliche Burst-Muster beinhaltet und uns eine Betrachtung darüber ermöglicht, wie das vorgeschlagene Verfahren auf diese reagiert und wann es zu kritischen Situationen kommt.

Blicken wir zunächst auf die Paketzwischenankunftszeiten $t_{A/B/C}$, um die Charakteristik der Bursts zu erfassen. Client B hat mit einem systematischen Burst-Verhalten zu kämpfen, der sich darin äußert, dass von ihm beim Server jeweils zwei Pakete unmittelbar nacheinander eintreffen – dieses dann dafür aber mit relativ konstanter Periode. Für den Datenfluss ist diese Eigenschaft gleichbedeutend mit einem Aufbau, in dem Pakete mit der doppelten nominellen Größe in Abständen von $2 \cdot t_\varphi$ übermittelt werden. Ein solches Verhalten kann in verschiedenen Szenarien eintreten, bspw. als nicht beeinflussbarer Teil der Kommunikation in Form von durch Interleaving verursachten Verzögerung. Ebenso kann es beabsichtigt als Ergebnis einer Re-Paketisierung auftreten, bspw. wenn die Audiohardware die Verarbeitung der nominellen Blockgrößen nicht

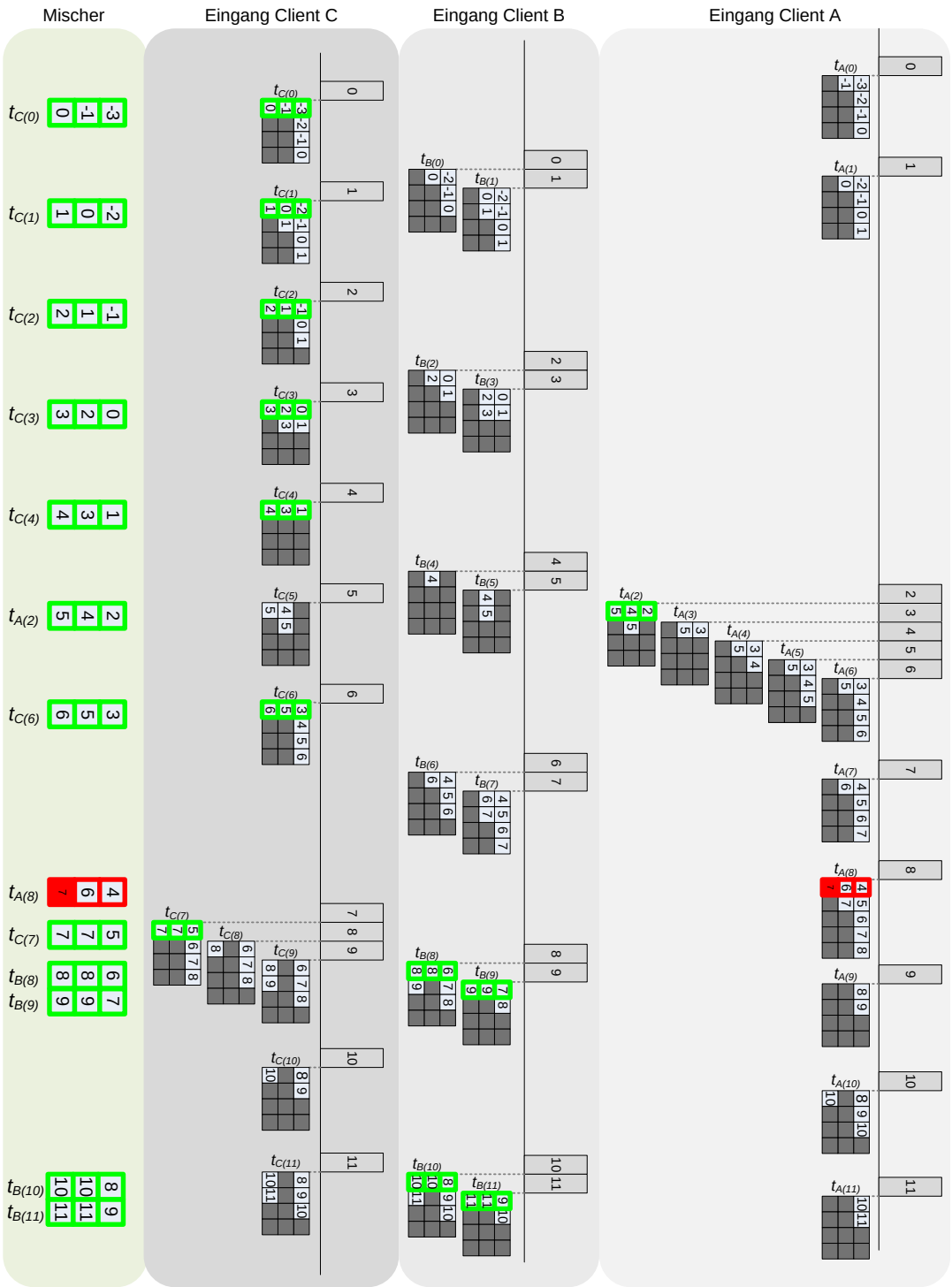

Abbildung 6.16: Zustand der Puffer Queue bei burstartigem Paketempfang

unterstützt: ein entsprechend eingeschränkter Client, der auf seine Audiohardware nur in Blöcken von 256 Samples zugreifen kann, teilt jeden Block auf zwei Audiopakete zu je 128 Samples auf und erhält damit ein für den NMP-Server transparentes Verhalten. Wir haben bereits in der Latenzanalyse festgestellt, dass sich eine solche Verringerung der Datengranularität nachteilig auf die Latenz auswirkt. In der nachfolgenden Untersuchung wird jedoch sichtbar, dass die Auswirkungen wegen der guten Periodizität nicht ins Gewicht fallen.

Anders als bei Client B treten bei A und C sporadische Burst-Muster auf, die Störungen repräsentieren, welche im Netz (z.B. bei Überlast) oder in den Endsystemen (bspw. verursacht durch variierende Interruptlatenzen) entstehen. Kennzeichnend sind Stauungen der Pakete über längere Zeiträume, die ein Vielfaches von t_φ betragen können, mit einem anschließenden schubartigen Empfang der gestauten Daten. Für Client A tritt nach $t_{A(1)}$ eine Stauung über mehrere Pakete auf, die zu einem Burst $(2,3,4,5,6)$ führt, während bei C nach einer deutlich kürzeren Stauung das Bündel $(7,8,9)$ entsteht.

Im Interesse der Übersichtlichkeit beginnen wir die Nummerierungen der Sequenzen für alle Clients neu bei Null. Zum Startzeitpunkt enthält q_a die Pakete ($p_{A(-3)}$, $p_{A(-2)}$ und $p_{A(-1)}$), q_b nur das Paket $p_{B(-1)}$, während q_c leer ist. Auf der Zeitachse trifft als erstes zu $t_{A(0)}$ Paket $p_{A(0)}$ ein und wird eingereiht, ohne einen Mischvorgang auszulösen. Unmittelbar darauf wird $p_{C(0)}$ empfangen, wodurch R1 erfüllt ist und das Paket $[-3,-1,0]$ gemischt wird. Wir haben somit zu Anfang unserer Betrachtung einen Versatzvektor von $\vec{s_v} = (-3,-1,0)$, der gleichbedeutend ist mit $\vec{s_v} = (0,2,3)$, da wir uns nur für die relativen Differenzen interessieren.

Im weiteren Verlauf treffen $p_{B(0)}$ und $p_{B(1)}$ unmittelbar nacheinander ein, wobei es aufgrund der leeren q_c zu keinem Mischen kommt. Das geschieht nach Erhalt von $p_{C(1)}$ mit der Generierung von $[-2,0,1]$. Diesem Mischvorgang schließt sich unmittelbar die erste gestörte Phase des Paketempfangs an, in der die Pakete $p_{A(2)}$ bis $p_{A(5)}$ gestaut sind. Beim vorliegenden Ausgangszustand der Puffer-Queues behindert die Störung unser Verfahren nicht, wie der Durchlauf durch die Abläufen nach Empfang der folgenden Pakete zeigt. Der begünstigende Umstand dabei ist, dass q_a zu diesem Zeitpunkt drei Pakete vorhält, die während der Stauung die Lücke teilweise überbrücken.

Über analytischer Betrachtung ist ersichtlich, dass eine von hier aus kritische Situation erst erreicht wird, wenn die Stauung bei A für mindestens sieben Pakete anhält, während die Pakete von B und C auch weiterhin ungestört eintreffen. Dabei würden die nächsten drei Pakete bei sukzessiver Entleerung von q_a gemischt werden. Bei weiter fortbestehender Stauung füllen sich q_b und q_c allmählich auf und führen zu einer Ausführung von R2, wenn der Stau durchgehend anhält, bis eine der beiden Puffer-Queues überläuft.

Eine solche kritische Situation wird in dieser Simulation bei der Stauung von A nicht erreicht. Wir sehen, dass die in q_a enthaltenen Pakete zu den Zeitpunkten $t_{C(2)}$, $t_{C(3)}$ und $t_{C(4)}$ abgebaut werden, nachdem die Pakete $p_{C(2)}$, $p_{B(2)}$, $p_{B(3)}$, $p_{C(3)}$ und $p_{C(4)}$ eintreffen und das Mischen der Pakete $[-1,1,2]$, $[0,2,3]$ und $[1,3,4]$ nach R1 erfolgt.

Nach dem Mischen bei $t_{C(4)}$ sind alle Puffer-Queues leer. Da die Stauung bei A auch weiterhin anhält, füllen sich mit dem Empfang von $p_{B(4)}$, $p_{B(5)}$ und $p_{C(5)}$ die Queues q_b und q_c. Unmittelbar vor dem Empfang des Paket-Bursts bei A gilt $q_b = \{4,5\}$ und $q_c = \{5\}$. Mit dem ersten Pakete im Burst $p_{A(2)}$ wird R1 aktiv und ermöglicht das Mischen von $[2,4,5]$. Damit ist q_a wieder leer und in der Lage, die unmittelbar hintereinander folgenden vier Pakete $p_{A(3)}$ bis $p_{A(6)}$ einzureihen und den Burst aufzufangen. Nach $t_{A(6)}$ erreichen die Puffer-Queues den zu Beginn unserer Betrachtung herrschenden Zustand mit $num_{p(A)} = 4$, $num_{p(B)} = 1$ und $num_{p(C)} = 0$. Auch der unveränderte Versatzvektor nach dem Mischen von $[3,5,6]$ bestätigt, dass diese relativ gewichtige Störung unproblematisch ist und keine Anwendung von R2 verursacht hat.

Dazu kommt es in diesem so konstruierten Verlauf an einer eher unscheinbaren Stelle: nach $t_{C(6)}$ stauen sich bei C lediglich die beiden Pakete $p_{C(7)}$ und $p_{C(8)}$, führen jedoch wegen der grundlegend anderen Ausgangslage zu einem Problem. Hinderlich sind dabei im Einzelnen folgende Punkte: erstens ist q_c leer;

zweitens ist q_a nahezu voll; und drittens läuft der Paketempfang von A und B während der Stauung ungestört weiter.

Betrachten wir den daraus resultierenden Ablauf genauer. Das zu $t_{A(7)}$ empfangene Paket kann q_a noch als letztes Element einreihen. Für die gemeinsam eintreffenden Pakete $p_{B(6)}$ und $p_{B(7)}$ ist die Situation unkritisch, beide werden in die vorher leere q_b untergebracht. Problematisch wird es mit dem Empfang von $p_{A(8)}$, für das kein Platz mehr in q_a vorhanden ist und aufgrund von R2 ein partielles Mischen von $[4,6,\emptyset(7)]$ erzwungen wird. Das erfolgt zum Zeitpunkt $t_{A(8)}$, unmittelbar nachdem $p_{A(8)}$ empfangen und anschließend in die frei gewordene Stelle in q_a eingefügt wird.

Bei diesem Zustand des Puffer-Queue Arrays beginnt ab $t_{C(7)}$ der Empfang der gestauten Pakete $p_{C(7)}$, $p_{C(8)}$ und $p_{C(9)}$. Die Art und Weise, wie das Einfügen dieses Paketbündels in q_c erfolgt, bestimmt die Funktionsfähigkeit unseres Verfahrens signifikant und soll daher näher untersucht werden.

Prinzipiell gibt es an dieser Stelle zwei Möglichkeiten, wie diese Pakete eingereiht werden: da das Paket $p_{C(7)}$ zum vorher durch R2 erzwungenen Mischen zum Zeitpunkt $t_{A(8)}$ nicht verfügbar war, können wir es, wie bei solchen Anwendungen üblich, als verspätetes Paket verwerfen, da es keiner sinnvollen Verwendung mehr zugeführt werden kann. Die andere Möglichkeit besteht darin, dieses Ereignis zu ignorieren und alle Pakete – inklusive des verspäteten – wie gewohnt einzureihen. Dieser unscheinbare Unterschied hat gewichtige Auswirkungen auf den Audiodatenstrom und die Charakteristik der Synchronisation, die wir im Anschluss an diese Simulation genauer betrachten werden. Hier sei vorweggenommen, dass wir uns in NMP für die zweite Variante entschieden haben und betrachten zunächst den restlichen Ablauf dieses Szenarios.

Sobald das erste Paket des ausstehenden Paketbündels von C eingereiht ist, ist R1 erfüllt und Paket $[5,7,7]$ kann gemischt werden. Die gleichzeitig empfangenen $p_{C(8)}$ und $p_{C(9)}$ werden in q_c eingereiht, da q_b zu diesem Zeitpunkt leer ist. Hier haben sich die Auswirkungen der Stauung bei C bereits abgebaut, es folgt ein zyklischer Ablauf bei dem A und C ihre Queues zunächst mit mindestens zwei Elementen füllen und auf Daten von B warten. Diese treffen dann mit dem beschriebenen Paketmuster in Zweierbündel ein und führen zu einem entsprechenden Muster der ausgehenden gemischten Pakete. So verlassen die Pakete $[6,8,8]$ und $[7,9,9]$ bzw. $[8,10,10]$ und $[9,11,11]$ unmittelbar nacheinander den Server zu den Clients. Damit ist wieder eine stabile Phase erreicht und wir beenden die Simulation dieses Szenarios.

Wir kommen zurück auf die ausstehende Analyse der Vorgänge rund um das innerhalb der Störung bei C durch R2 erzwungene partielle Mischen. Dabei sind zwei Aspekte genauer zu untersuchen: zunächst müssen wir uns vor Augen halten, welchen Effekt das Fehlen eines Teils der Audiodaten auf die Anwendung hat. Anschließend sind die unterschiedlichen Auswirkungen der beiden angesprochenen Möglichkeiten, wie mit verspäteten Paketen zu verfahren ist, auf die Wahrnehmung beim Musiker genauer zu untersuchen.

Die Betrachtung des ersten Punktes ist dabei schnell abgeschlossen, denn die Auswirkungen von Lücken im kontinuierlichen Datenstrom sind aus vergleichbaren Anwendungen aus dem Audiobereich bekannt: im zum Zeitpunkt $t_{A(8)}$ versendeten Paket $[4,6,\emptyset(7)]$ fehlen die korrespondierenden Daten von Client C, wodurch es zu einer Diskontinuität kommt, die unweigerlich zu einer teils heftigen Störung des wahrgenommenen Audiosignals führt. Diese Störung beschränkt sich nicht auf dem Client, dessen Audiodaten zum fraglichen Zeitpunkt nicht verfügbar sind, sondern betrifft alle Teilnehmer, da die Datenlücke an alle verteilt wird. Hier kommen dann die in Abschnitt 6.3 vorgestellten Verfahren zur Fehlerverdeckung zum Tragen, die eine Abmilderung von subjektiv wahrgenommenen Auswirkungen solcher Störungen umsetzen.

Für die beim erzwungenen Mischen fehlenden Pakete haben wir die beiden Möglichkeiten der Weiterverarbeitung bereits kurz angesprochen. Das Verständnis der strengen Sequenzierung gebietet es, die verspäteten Pakete zu verwerfen. Üblicherweise wird in typischen Audioanwendungen die Entscheidung, ob ein Paket für die Verarbeitung verfügbar ist, unmittelbar vor der Übertragung der Daten an die Audiohardware getroffen. Offensichtlich sind dann Pakete, die nach diesem Abspielzeitpunkt empfangen werden, wertlos.

Anders ist es beim NMP-Server, denn diese Komponente liegt in etwa in der Mitte des Datenpfades aller Audiopakete und bedingt dadurch, dass verspätet empfangene Pakete für die Clients durchaus noch von Nutzen sein können. Über das ultimative Kriterium, wann Daten wertlos werden, verfügen nur die Clients, so dass eine Weiterleitung veralteter Pakete sehr wohl sinnvoll ist.

Unter der Vorgabe, dass die Clients letztlich die Verarbeitung und Ausgabe der vom Server erhaltenen Daten kontrollieren, erscheint die Beibehaltung der üblichen Strategie mit einer zusätzlichen Übermittlung verspäteter Pakete naheliegend zu sein. In unserer Simulation würde das bedeuten, dass das verspätete Paket $p_{C(7)}$ nicht in die Warteschlange eingereiht, sondern direkt an alle Clients versendet wird, die dann selbständig über die weitere Verwendung der Daten entscheiden können. Anschließend würde $p_{C(8)}$ als erstes Paket des Bursts regulär verarbeitet und zu [5,7,8] gemischt werden. Der Versatzvektor würde sich bei dieser Variante nicht ändern und über die gesamte Sitzung einen theoretisch idealen konstanten Wert beibehalten.

Real haben wir es dagegen mit Clients mit potentiell untereinander abweichenden Sampleraten zu tun, die im Ergebnis zu einer kontinuierlichen Drift des Versatzvektors führen. Unser gewählter Ansatz, verspätete Pakete statt zu verwerfen regulär zu verarbeiten, trägt dieser Erwartung Rechnung und ermöglicht die Kompensation von Abtastratenunterschieden über die Datensynchronisation. Wir werden in einem eigenen Simulationsdurchlauf diesen Aspekt weiter unten genauer untersuchen und auch die Auswirkungen dieses Versatzes auf die Teilnehmer erläutern.

Davor müssen wir an dieser Stelle klären, wie kritisch Bursts für das vorgestellte Verfahren sind und in welchem Umfang sie kompensiert werden können. Wir wollen diesen Nachweis analytisch führen und bestimmen die konstante Paketrate als unsere Basis, wobei wir hier die Taktratenunterschiede ausklammern und von einer einheitlichen Paketrate von 375 Hz ausgehen. Wir wissen, dass Netz- und Verarbeitungs-Jitter zu erheblichen Variationen einzelner Zwischenankunftszeiten führen können, deren extreme Auswirkungen sich in Stauungen und Bursts manifestieren. Diese beiden Größen sind voneinander direkt abhängig: jedem Burst muss eine entsprechende Stauung vorher gegangen sein, während umgekehrt eine Stauung zu einem Burst führen muss. Dabei ist zu beachten, dass wir Lücken bei der Übertragung aufgrund von verworfenen Paketen indirekt als Stauung betrachten und die Länge der Lücke anhand der Sequenznummern berechnen können.

Im vorgestellten Verfahren gibt es nur zwei Szenarien, in dem es zu Überläufen und zur Anwendung von R2 kommt: 1) ein Client X hat eine Stauung, die länger anhält, als die übrigen Clients einkommende Pakete puffern können, oder 2) ein Client X erhält in einem Burst mehr Pakete, als in seiner Queue q_X Platz haben. Der zweite Fall impliziert, dass vor dem Burst eine Stauung aufgetreten ist und kann auf diese zurückgeführt werden. Wir beschränken uns daher auf die Betrachtung der Vorgänge während einer Stauung.

Im ersten Fall ist ein kritischer Zustand dann erreicht, wenn die Stauung so lange anhält, bis die in q_X enthaltenen Pakete konsumiert wurden und eine der übrigen Queues überläuft. Der denkbar ungünstigste Ausgangszustand für eine solche Situation ist der, dass unmittelbar vor der Stauung q_X leer und mindestens eine Queue q_Y voll ist, es also gilt $num_{p(Y)} = 4, num_{p(X)} = 0, \Delta num_{p(XY)} = 4$. Für jedes der während der Stauung von Client Y eintreffenden n_{p_S} Pakete wird R2 ausgeführt und unvollständig gemischt. Wenn die gestauten Pakete von X anschließend als Burst eintreffen, ist q_X leer und die Pakete können eingereiht bzw. über Richtlinie R1 direkt mit bereits vorhandenen Paketen der anderen Clients gemischt werden. In jedem Fall ändert sich nach der Abarbeitung des Bursts der stabile Zustand dahin gehend, dass sich die Differenz der Füllgrade von q_X und q_Y um n_{p_S} verringert, es gilt dann $\Delta num_{p(XY)} = 4 - n_{p_S}$.

So kritisch die unvollständigen Mischvorgänge für die Audioqualität sind, für die künftige Synchronisation ist das Ergebnis sehr positiv: die Differenz des höchsten und des niedrigsten Füllstands hat sich verringert und damit die mittleren Füllgrade fortan angeglichen. Der anfängliche kritische Zustand wurde behoben, der erreichte stabile Zustand ist in der Lage, künftig Stauungen bei X von bis zu n_{p_S} Paketen zu kompensieren.

Diese Verbesserung ist relativ zum initialen Zustand zu verstehen, in dem die Füllstände der Queues eine Stauung bei X nicht zugelassen haben. Nach der Anpassung ist somit zwar die Gefahr eines Überlaufs bei künftigen Stauungen von X reduziert, gleichwohl hat sie aber dazu geführt, dass für andere Clients kritische Zustände wahrscheinlicher geworden sind. Dadurch, dass Anpassungen kontinuierlich nach dem aktuellen Bedarf vorgenommen werden, ist einerseits sicher gestellt, dass in der Initiierungsphase bereits nach sehr kurzer Laufzeit ein optimaler Füllstand der Queues erreicht wird, der andererseits fortlaufend an sich ändernde Paketeingangsmuster angepasst wird und so zu jeder Zeit höchstmögliche Robustheit bietet.

Gleichwohl lassen sich Szenarien konstruieren, die eine Oszillation des Verfahrens verursachen können. Am einfachsten nachvollziehbar ist dabei ein Aufbau, bei dem unterschiedliche Clients Stauungen und Bursts in dem Umfang haben, der die Größe der Warteschlangen übersteigt. Kommen von diesen die Daten bspw. jeweils in Schüben von sechs Paketen an, müssen kontinuierlich Pakete verworfen und der Versatz korrigiert werden. Generell sind oszillierende Szenarien immer solche, die die Kompensationsfähigkeit der Warteschlangen ohnehin übersteigen und mit einer so dimensionierten maximalen Verweildauer im Server nicht realisierbar sind. Wir haben uns hierbei darauf beschränkt, solche Szenarien anhand der Häufigkeit der Anpassungen zu erkennen und den Dienst abzubrechen. Unsere Erweiterungen dieses Grundverfahrens, die im nächsten Abschnitt beschrieben werden, decken u.A. diese Schwachstelle ab.

An dieser Stelle können wir nun auch die im vorigen Abschnitt aufgekommene Frage klären, warum eine vorauseilende Anpassung der Füllstände der Queues schädlich ist. Erinnern wir uns dafür noch einmal an den erreichten Zustand nach abgeschlossener Initiierung beim Übergang in die stabile Phase: wir haben gesehen, dass die Anwendung der vorgestellten Richtlinien immer dazu führt, dass beim Übergang die Queue eines Clients (nämlich des als erstes beigetretenen) vollständig gefüllt ist, während die des zuletzt beigetretenen Clients leer bleibt, da jedes von ihm empfangene Paket einen Mischvorgang auslöst und sofort verbraucht wird. Der vorgebrachte Einwand bezog sich anschließend darauf, dass ein solcher Füllstand der Queues für ein De-Jitter Puffer nicht optimal ist. Mit den dargestellten Eigenschaften bei Datenstauungen und -Schüben wissen wir nun allerdings, dass kritische Situationen zwar beim ersten Auftreten zu Störungen führen, danach die Füllstände jedoch automatisch derart angepasst werden, dass solche Ereignisse künftig toleriert werden können.

Bleiben wir zur besseren Veranschaulichung bei unserem Beispiel aus der Darstellung 6.15: kritisch ist hierbei, wenn Stauungen bei Client B oder C auftreten und gleichzeitig die Pakete von A weiterhin eintreffen. Die Anwendung der PUSH Richtlinie sorgt während dieser Stauung dafür, dass über einen Paketversatz die Füllstände angeglichen werden, so wie es in dieser Simulation deutlich geworden ist. Der Versatz entspricht dabei der Anzahl verspäteter Pakete und führt dazu, dass für den folgenden Ablauf der Sitzung für diesen Client eine entsprechende Reserve aufgebaut wird.

Ein dauerhaft hoher Füllstand einer Queue birgt somit einen kritischen Zustand für den Fall, wenn es bei einem der anderen zu einer Stauung kommt. Gleichzeitig bedeutet er jedoch auch, dass eine Stauung bei eben diesem Client in einem viel höheren Ausmaß toleriert werden kann. Im beschriebenen anfänglichen Füllstand der Queues kann so bspw. bei A eine Stauung von acht Paketen toleriert werden, ohne dass es zu einer Störung des Datenflusses kommt. Man stelle sich dazu eine Konstellation vor, bei der die Audiopakete dieses Client in regelmäßigen Abständen für die Dauer von 20 ms verzögert werden, dann ist ein permanenter Füllstand von vier Paketen die einzige Möglichkeit, einen solchen Client störungsfrei an einer Sitzung teilnehmen zu lassen. Eine äußere Anpassung der Füllstände hätte dann fatale Folgen für das Musizieren.

Im Ergebnis bedeutet das, dass eine vorauseilende Korrektur der Füllstände aus mehreren Gründen nicht sinnvoll ist: erstens wäre sie nur dann von Vorteil, wenn a priori Kenntnisse über Art und Umfang von Stauungen vorhanden wären, was ihren zufälligen Charakter widerspricht. Zweitens verursacht eine vorauseilende Versatzanpassung genau die gleichen Störungen im Datenstrom, wie eine über die PUSH Direktive

erzwungene, womit sich für die proaktive Variante kein Vorteil ergibt. Und nicht zuletzt wird durch eine vorauseilende Korrektur dem De-Jitter Puffer die Fähigkeit genommen, Clients mit längeren Stauungen zu unterstützen. Ergo ist eine vorauseilende Korrektur der Füllstände nicht nur unnötig, sondern auch schädlich.

Zusammenfassend sind bei der Behandlung von Stauungen und Schüben folgende wesentliche Merkmale unseres Verfahrens zu benennen: es stellt sicher, dass ungünstige Zustände der Queues mit dem ersten Überlauf – also erst dann, wenn es unvermeidlich ist – angepasst und verbessert werden. Damit wird die Anzahl erforderlicher Anpassungen minimiert und zielt darauf ab, die mittleren Füllstände den aktuellen Paketankunftsmuster entsprechend zu optimieren. Damit ist das Verfahren inhärent stabilisierend und adaptiv optimierend.

6.7.5.4 Behandlung unterschiedlicher Paketraten

Aus Abschnitt 6.6 wissen wir, dass verteilte Clients unvermeidbar mit voneinander unterschiedlichen Audioabtastraten betrieben werden und zu unterschiedlichen Paketraten führen. Die Fähigkeit, mit diesen Abweichungen umgehen zu können, haben wir daher einleitend als eine Grundvoraussetzung für unser Synchronisationsverfahren definiert. In diesem Abschnitt sollen die Auswirkungen ungleicher Paketraten auf das vorgestellte Verfahren anhand einer eigenen Simulation untersucht werden.

Mit den im letzten Abschnitt ermittelten Abschätzungen für die zu erwartenden Gangabweichungen bei den Sampleraten im ‰-Bereich können wir keine sinnvolle Darstellung eines solchen Ablaufs erreichen, weswegen wir für die hier angestellte Untersuchung ein realitätsfernes Szenario gewählt haben: wie gehabt sind die Paketzwischenankunftszeiten der Clients A, B und C und die Verarbeitung beim Server in Abbildung 6.17 dargestellt. Wir wollen uns bei dieser Betrachtung allein auf die Drift-Problematik beschränken und haben Clients B und C deshalb mit idealen Paketraten und Zwischenankunftszeiten von t_φ versehen, wohingegen Client A eine um etwas mehr als 50 % höhere Paketrate von 570 Hz hat.

Auf eine detaillierte Betrachtung aller Einzelschritte verzichten wir hier, da der interessierte Leser diese aufgrund der bisher durchlaufenen Simulationen selbst nachvollziehen kann. Stattdessen soll anhand der relevanten Stellen auf die Eigenschaften der dem Verfahren innewohnenden Kompensationsfähigkeit eingegangen und die Auswirkung auf die Wahrnehmung bei den Beteiligten eingegangen werden.

Beginnen wir diese Untersuchung mit den aus der bisherigen Betrachtung gewonnen Erkenntnissen erwarteten Verhalten. Ein Client mit einer systematisch höheren Paketrate wird dazu führen, dass dessen Warteschlange beim Server zwangsläufig nach einiger Zeit überläuft, da die aufgrund der PULL Richtlinie erreichte Verbrauchsrate geringer ist als die Produktionsrate dieses Clients. Beim Überlauf seiner Queue erzwingt dieser Client mit der PUSH Regel ein partielles Mischen und gleicht über eine Driftanpassung den Füllstand seiner Queue an die der übrigen an.

Genau dieses Verhalten führt die Simulation dieses Szenarios hier zutage, wobei wir zwei unterschiedlich Ausprägungen beobachten können. Der erste Überlauf tritt zum Zeitpunkt $t_{A(5)}$ auf. q_A ist bereits voll und empfängt das nächste Paket, q_B ist leer und q_C hat nur das Kopfelement $p_{C(3)}$. Aufgrund von R2 wird das Paket $[1,\emptyset(3),3]$ gemischt. Das Paket $p_{B(3)}$ trifft erst nach dem erzwungenen Mischvorgang ein und wird in q_B eingereiht. Noch bevor $p_{C(4)}$ verfügbar ist, erzwingt der Empfang von $p_{A(6)}$ erneut ein partielles Mischen des Paketes $[2,3,\emptyset(4)]$. Nach Eingang von $p_{C(4)}$ und $p_{B(4)}$ kann über PULL das Paket $[3,4,4]$ regulär gemischt werden. Wir erkennen, dass sich der anfängliche Versatzvektor $\vec{s_v} = (0,2,2)$ verändert hat und nun $(0,1,1)$ beträgt – die beabsichtigte Anpassung des Versatzes hat stattgefunden.

Dieser Ablauf wiederholt sich unmittelbar danach mit dem partiellen Mischen von $[4,\emptyset(5),5]$ und $[5,5,\emptyset(5)]$, gefolgt vom vollständig gemischten Paket $[6,6,6]$. Im Ergebnis führt auch diese Runde zu einer Versatzanpassung zu $\vec{s_v} = (0,0,0)$.

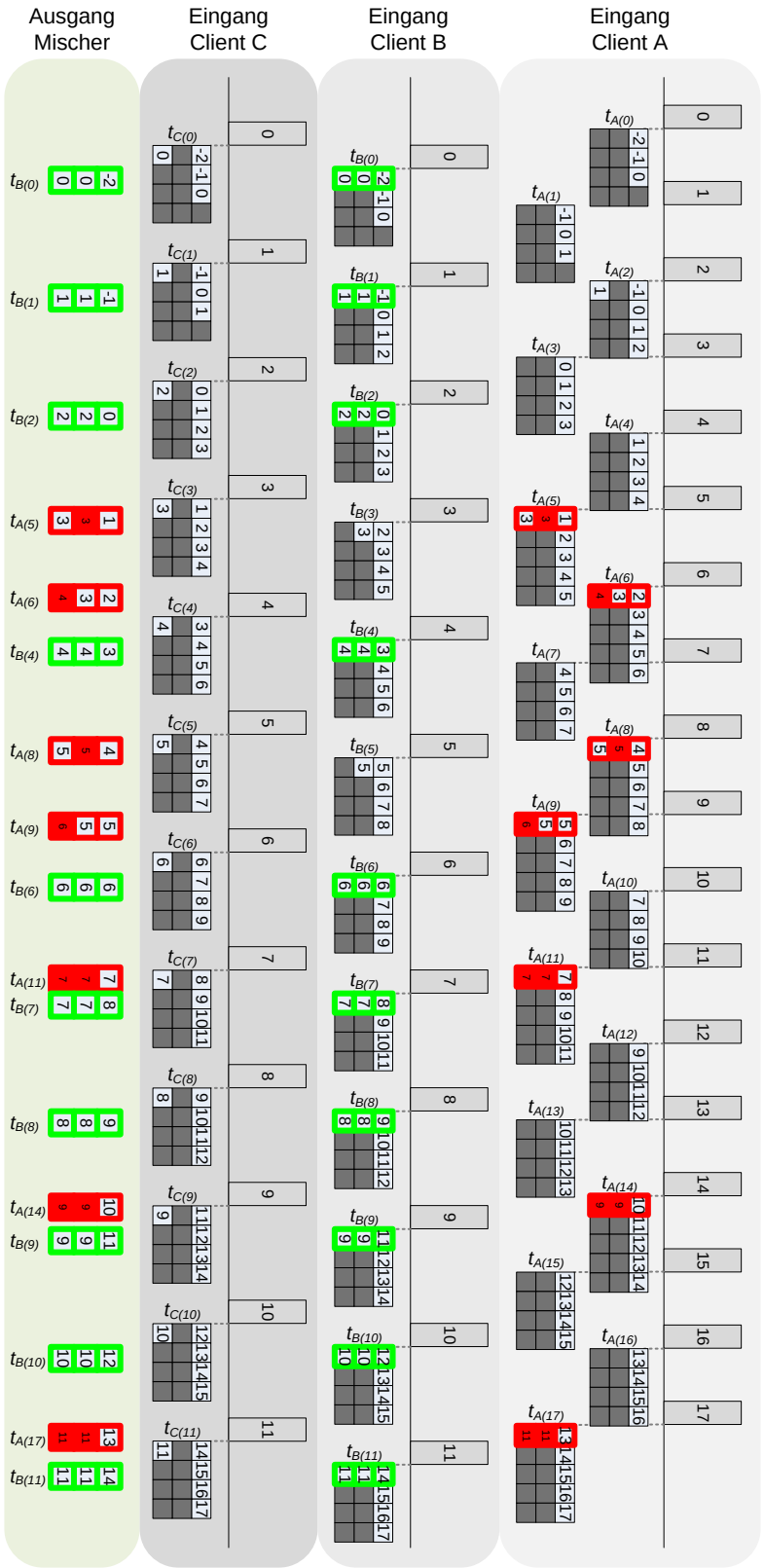

Abbildung 6.17: Auswirkungen unterschiedlicher Paketraten auf die Synchronisation

Diesem Ereignis folgt anschließend die zweite Ausprägung der Anpassung: die so gewählte Paketratendifferenz sorgt für einen Versatzdrift derart, dass zum Zeitpunkt $t_{A(11)}$ q_A überläuft, während q_B und q_C leer sind. Das führt zu einem erzwungenen Mischen von $[7,\emptyset(7),\emptyset(7)]$, einem Paket also, in dem nur die Audiodaten von Client A vorhanden sind. Ein solches Paket kann vom Server verworfen werden, da eine Übertragung für keinen Client verwertbare Informationen enthält. Nach Erhalt der korrespondierenden Pakete $p_{C(7)}$ und $p_{B(7)}$ kann anschließend $[8,7,7]$ vollständig gemischt werden. Es folgen weitere Runden dieses Ablaufs, in denen sich zwei PULL- und ein PUSH-Paket abwechseln.

In dieser Ausprägung ist die Driftanpassung deutlicher sichtbar, da sie sich darin auswirkt, dass die Daten des Clients mit zu hoher Paketrate direkt verworfen werden. Das Muster von zwei vollständig und einem unvollständig gemischten Paket entspricht genau der um knapp 50 % erhöhten Paketrate, die mit einer ebensolchen Verwurfsrate kompensiert wird.

Die funktionale Anforderung erfüllt unser Verfahren damit, es bleibt im Folgenden zu klären, welche Auswirkungen die Driftanpassung auf die Clients hat. Blicken wir dazu zunächst auf die zweit genannte Ausprägung: in unserer Simulation verwirft der Server die Pakete $[7,\emptyset(7),\emptyset(7)]$, $[10,\emptyset(9),\emptyset(9)]$ und $[13,\emptyset(11),\emptyset(11)]$. Betrachten wir stellvertretend für alle den Ablauf bei den Clients für das erste verworfene Paket. Client A empfängt vom Server die Pakete $[6,6,6]$ gefolgt von $[8,7,7]$ und kann anhand der Sequenznummern erkennen, dass a) sein Paket mit der Sequenznummer 7 nicht verarbeitet wurde und b) eine Anpassung des Versatzvektors stattgefunden hat. Er fügt nun das von ihm zum Server versendete Paket $p_{A(7)}$ in die Fehlstelle ein, so dass der an die Audiohardware übergebene Datenstrom aus der Paketfolge $[6,6,6]$, $[7,\emptyset(7),\emptyset(7)]$ und $[8,7,7]$ besteht. Damit stellt A – unabhängig von der vom Server empfangenen Paketrate – sicher, dass die an die Audiohardware übergebene und die von dort ausgelesene Datenmenge identisch ist.

Auch Clients B und C erhalten $[6,6,6]$ gefolgt von $[8,7,7]$ und erkennen die Versatzänderung. Da jedoch die auszugebende Sequenznummer der erwarteten entspricht, können die empfangenen Pakete direkt ausgegeben werden. Die geschriebene Datenrate entspricht dabei genau der gelesenen.

Im Ergebnis genügt unser Verfahren in dieser Ausprägung somit die eingangs als einzig sinnvoll genannte Maßnahme zur Kompensation: Client mit höherer Abtastrate strecken den vom Server empfangenen Datenstrom durch einfügen zusätzlicher Pakete, während die langsameren den Datenstrom durch Verwurf von Paketen stauchen.

Der ersten Ausprägung der Driftanpassung zwischen $t_{A(3)}$ und $t_{B(6)}$ ist dagegen etwas schwerer anzusehen, dass das beabsichtigte Verhalten erreicht wird. Zur Verdeutlichung betrachten wir auch hier stellvertretend die Ereignisse bei den Clients beim Empfang der gemischten Pakete $[1,\emptyset(3),3]$, $[2,3,\emptyset(4)]$ und $[3,4,4]$. Client A erkennt, dass die Pakete $[1,\emptyset(3),3]$ und $[2,3,\emptyset(4)]$ unvollständig gemischt wurden. Gleichzeitig entsprechen die empfangenen Sequenznummern den erwarteten, so dass die Pakete direkt ausgegeben werden können. In diesem Fall streckt somit der Server durch das partielle Mischen den Datenstrom der langsameren Clients.

Clients B und C müssen dagegen die empfangenen Pakete weiterverarbeiten, um aus diesen die für sich relevantesten Daten zusammenzustellen, so wie folgend beispielhaft für B erläutert wird. Mit dem Empfang von $[1,\emptyset(3),3]$ kann B nicht feststellen, warum seine Daten im gemischten Paket nicht enthalten sind. $p_{B(3)}$ kann einerseits bei der Übertragung zum Server verloren gegangen sein, andererseits kann B selbst eine abweichende Paketrate haben, die der Server im Audiodatenstrom mit dieser Lücke streckt. Abhängig vom Zustand des Empfangspuffers kann B nun entweder auf das nächste Paket vom Server warten, um darüber Gewissheit zu erlangen. Muss dagegen das empfangenen Paket sofort in die Audiohardware geschrieben werden, kann B es vor der Ausgabe mit seiner lokal vorhandene Kopie von $p_{B(3)}$ zu $[1,3,3]$ vervollständigen. Damit werden Interpretationen und allfällige Korrekturen auf das nächste verschoben.

Kann andernfalls B auf das Eintreffen des nächsten Paketes warten, ist mit Empfang von $[2,3,\emptyset(4)]$ die

Situation eindeutig: es ist gesichert, dass das Fehlen von $p_{B(3)}$ im vorhergehenden Paket nicht aufgrund von Übertragungsverlusten verursacht wurde. Gleichzeitig steht fest, dass das Paket nicht verworfen wurde, um eine geringe Paketrate zu kompensieren, da in diesem Fall im zuletzt empfangenen Paket $p_{C(4)}$ enthalten sein müsste. Dass das Fehlen nicht auf einen Paketverlust zurückzuführen ist, wird spätestens beim Empfang von $[3,4,4]$ deutlich, denn der geänderte Versatzvektor bestätigt, dass die Anpassung aufgrund der höheren Paketrate von A erfolgt ist. Mit dieser Information kann B nun die auszugebenden Audiodaten so rekonstruieren, dass für Musiker und Zuhörer die geringst mögliche Störung der Wahrnehmung erfolgt.

Da Client B weiß, dass im Datenstrom von A ein Paket zu verwerfen ist, muss zwischen $[0,2,2]$ und $[3,4,4]$ ein Paket $[X,3,3]$ eingefügt werden, wobei X aus $p_{A(1)}$, $p_{A(2)}$ oder einer günstigen Kombination beider Pakete bestehen sollte. Wie die Erzeugung eines solchen Paketes zur Reduzierung der wahrnehmbaren Störung erfolgen kann, wurde genauer in Abschnitt 6.3 diskutiert. Hier beschränken wir uns darauf zu zeigen, dass das erforderliche Paket aus den verfügbaren Daten gewonnen werden kann. Aus Paket $[2,3,\emptyset(4)]$ kann B das Paket $p_{A(2)}$ rekonstruieren, da das eigene Paket $p_{B(3)}$ und sein Anteil am gemischten Paket bekannt sind und $p_{A(2)}$ so durch Subtraktion gewonnen werden kann. Im einfachsten Fall ist B dann in der Lage, das partielle Paket $[1,\emptyset(3),3]$ mit seinen Daten von $p_{B(3)}$ zu vervollständigen und auszugeben.

Wir können leicht zeigen, dass die hier dargestellte perfekte Rekonstruktion der vom Server verworfenen Daten bei drei Clients immer erreicht werden kann. In diesem Fall gibt es für ein unvollständiges Mischen nur die zwei Möglichkeiten, dass im Mischpaket die Daten von einem oder zwei Clients enthalten sind. Wie oben gezeigt, werden allerdings Mischpakete, die nur die Daten eines Clients enthalten, nicht verarbeitet, da sie keine verwertbaren Informationen enthalten. Für einen Client B sind daher nur die beiden Möglichkeiten eines unvollständigen Mischens zu berücksichtigen, bei denen entweder das Mischpaket die Daten von B enthält oder nicht. Im ersten Fall kann B mit Hilfe der lokal gespeicherten Kopie des Paketes mit der korrespondierenden Sequenznummer die Audiodaten des zweiten Paketes extrahieren und für evtl. Weiterbearbeitung vorhalten. Im zweiten Fall kann das Fehlen der eigenen Audiodaten im partiell gemischten Paket einfach aus der Kenntnis der Mischparameter und dem lokalen Vorhandensein der erforderlichen Daten das Mischpaket vervollständigt werden.

Diese perfekte Rekonstruktion versagt bei mehr als drei Clients, da dann keine eindeutige Zuordnung der im Mischpaket enthaltenen Audiodaten und damit die Umkehr des Mischvorgangs beim Client mehr möglich ist. Aus diesem Grund ist ein proaktives Verwerfen einzelner Pakete beim Server (so wie in unserem Beispiel mit $[7,\emptyset(7),\emptyset(7)]$, $[10,\emptyset(9),\emptyset(9)]$ und $[13,\emptyset(11),\emptyset(11)]$ erfolgt) anzustreben. Diese sind mit der Erweiterung unseres Basisverfahrens möglich, die wir im nächsten Abschnitt diskutieren werden.

Hier bleibt zusammenfassend festzuhalten, dass das vorgestellte Verfahren in der Lage ist, die in einer NMP-Sitzung aufgrund von unterschiedlichen Abtastraten auftretenden Variationen der Paketraten zu kompensieren. Dies erfolgt in Abhängigkeiten von den Zuständen der Puffer-Queues entweder durch direkten Verwurf von Paketen der schnelleren Clients beim Server, oder durch die Neu-Komposition der Audiodaten aus partiell gemischten Paketen beim Client, wo überschüssige Daten ebenso verworfen werden. Im Ergebnis gibt jeder Client die gleiche Anzahl von Paketen an seine Audiohardware, wie er von dieser ausliest. Die Kompensation der Paketratenunterschiede resultiert in eine kontinuierliche Anpassung des Versatzvektors, die vom Server für die synchronisierte Ausgabe der archivierten Audiospuren zu protokollieren ist.

Abschließend sei an dieser Stelle noch einmal erwähnt, dass das demonstrierte Beispiel weit abseits der Szenarien liegt, die wir aufgrund unterschiedlicher Sampleraten üblicherweise zu kompensieren haben. Zwar können wir die Funktionalität des Verfahrens mit einem Aufbau, in dem ein Client mit 48 kHz und zwei mit 32 kHz arbeiten, auch praktisch nachweisen. Verständlicherweise kann dabei aufgrund des hohen Grades an Diskontinuität im Audiodatenstrom keine sinnvolle Verwendung des Systems geboten werden.

6.7.5.5 Bewertung und Fazit

In diesem Abschnitt haben wir ein reaktives, höchst einfaches und gleichzeitig effizientes Verfahren für die Synchronisation multipler Audiodatenströme vorgestellt und anhand von Simulationen und analytischen Untersuchungen nachgewiesen, dass es allen Anforderungen an unserem NMP-Server genügt.

Es basiert auf ein Array statischer Puffer-Queues für jeden teilnehmenden Client und lediglich zwei Richtlinien, wann die empfangenen Pakete zu synchronisieren und zu mischen sind. Die Entscheidungsfindung wird bei jedem Empfang eines Audiopaketes im Server vorgenommen und basiert ausschließlich auf den Momentanzustand der Queues, die anhand der Sequenznummern empfangener Pakete befüllt werden. Dadurch ist das Verfahren zustandslos und kommt ohne jegliche Zeitbasis aus.

Das vorgestellte Verfahren ist in der Lage, Stauungen und Schübe beim Paketempfang zu kompensieren, solange diese innerhalb der durch die statische Länge der Queues gegebenen Reserve bleiben. Wird diese Reserve überschritten, erfolgt anhand des Paketeingangsmusters eine Adaption der Reserve über eine Versatzanpassung derart, dass von Clients mit höherem Jitter eine größere Paketanzahl in den Queues vorgehalten wird. Diese Anpassung erfolgt permanent und führt zu einem zu jedem Zeitpunkt robustest möglichen Füllstand des Puffer-Arrays – das Verfahren ist selbst optimierend.

Für die bei Zusammenspiel unterschiedlicher Clients immanent vorhandene Abweichung der Paketraten vom Sollwert wurde das Verwerfen ganzer Audiopakete als einzig sinnvolle Methode zur Kompensation ermittelt, mit dem das Stauchen von Audiodatenströmen schneller Clients umgesetzt wird. Unser Verfahren genügt dieser Forderung und setzt diese Anpassung auf zwei vom Füllstand der Queues abhängige Arten um. Das direkte Verwerfen von Paketen des schnellsten Clients entspricht genau der anvisierten Strategie. Im anderen Fall erfolgt ein partielles Mischen und der Übertragung der Kompensationsarbeit auf die Clients.

Mit diesen Eigenschaften entspricht dieses Verfahren dem Ideal, hoher Funktionalität bei größtmöglicher Einfachheit. Der entsprechend implementierte NMP-Server zeichnet sich aufgrund dieser Einfachheit durch höchste Robustheit und Stabilität aus, die sogar die in diesem Abschnitt zur Demonstration weit abseits regulärer Parameter entworfenen Szenarien unterstützt.

6.7.6 Proaktive Synchronisation

So zuverlässig und funktional dieses simple Basisverfahren ist – mit einer relativ geringfügigen Änderung können wir einige der beobachteten und diskutierten Mängel beheben. Diese Erweiterung hin zu einem proaktiven Ansatz ist Inhalt dieses Abschnitts. Grundlage dafür ist eine zeitliche Zustandsprotokollierung, die über den Momentanzustand der Puffer-Queues hinaus kontinuierliche Messungen vornimmt und damit hilfreiche Rückschlüsse auf Kenngrößen liefert.

6.7.6.1 Schwächen und Verbesserungspotenziale des reaktiven Ansatzes

Bevor wir die Erweiterungen beschreiben, listen wir zunächst die dem reaktiven Basisverfahren innewohnenden Schwächen auf, deren Behebung wir anstreben.

Abhängigkeit der Pufferstände vom Beitrittszeitpunkt zur Sitzung

Bei der Simulation und Diskussion einer Aufbauphase in Abschnitt 6.7.5.2 haben wir festgestellt, dass beim Übergang in die stabile Phase der Sitzung der Zustand der Warteschlangen ganz wesentlich davon abhängt, in welcher Reihenfolge die Sitzung aufgebaut wurde. Im Ergebnis ist dabei die Queue des zuerst beigetretenen Client im Mittel voll, während die des letzten leer ist. Das führt dazu, dass die Daten des ersten eine sehr hohe Verweildauer im Server haben, während die des letzten praktisch immer sofort gemischt und zurückgesendet werden. Das ist in dem Fall, wenn alle Teilnehmer die glei-

che Abspielverzögerung verwenden, nicht relevant, da der Abspielzeitpunkt dann abhängig von der Verweildauer im Server ist.

Anders verhält es sich dagegen, wenn die Clients die Abspieldauer dynamisch an die Netzeigenschaften und den Nutzereinstellungen anpassen. Dann ist der zuerst beigetretene Teilnehmer benachteiligt, da seine Daten in jedem Fall am längsten beim Server liegen. Zwar bedeutet ein hoher mittlerer Füllgrad seiner Warteschlange für diesen Client auch einen besseren Schutz vor Stauungen – die dieser Client jedoch vielleicht gar nicht braucht.

Eine dynamische Anpassung der Abspielverzögerung beim Client ist somit nur dann sinnvoll, wenn auch der Server über entsprechende Mechanismen verfügt. Er muss in der Lage sein, den relativen Versatz der Audiodatenströme zueinander vorausschauend zu modifizieren, ohne auf eine erzwungene Anpassung über die PUSH Richtlinie zu warten.

Statische Dimensionierung der Warteschlange führt zu Worst Case Latenz

Verfahrensbedingt verursachen Unregelmäßigkeiten bei der Datenübertragung einen stetigen Anstieg der Füllstände der Warteschlangen und damit eine Erhöhung der mittleren Verweildauer der Pakete auf den maximal zulässigen Wert. Das ist einfach nachvollziehbar anhand der Wirkungsweise der verwendeten Richtlinien: PULL kommt bei regulär eintreffenden Daten zum tragen, während PUSH für die Korrekturen sorgt. Damit Missstände also behoben werden, müssen die betreffenden Queues überlaufen.

Auch hierbei gilt, dass diese Eigenschaft des Basisverfahrens keine Relevanz hat, solange die Clients mit statischen Abspielverzögerungen arbeiten. Um Audiodaten darüber hinaus tatsächlich auch nur so lange im Server zu belassen, wie für eine fehlerfreie Synchronisation nötig, muss der Server unnötig hohe Füllstände erkennen und reduzieren können.

Anpassung von Paketratenunterschieden führt meist zu partiellem Mischen

Wie weiterhin ermittelt, erfolgt die Korrektur von Paketratenunterschieden durch Verwurf von Daten des schnellsten Clients. Da dies reaktiv und über die PUSH Regel erfolgt, kann dieser Verwurf auch über mehrfache partielle Mischvorgänge erfolgen. Diese erhöhen einerseits das Datenaufkommen und führen andererseits bei den Clients zu einem erhöhten Rekonstruktionsaufwand bzw. behindern eine vollständige Wiederherstellung der originalen Audiodatenströme.

Mit der Fähigkeit, eine erhöhte Paketrate detektieren zu können, bevor diese zu einem Überlauf der Warteschlange führt, wäre der Server in der Lage, Pakete vorausschauend zu verwerfen und die Kompensation unterschiedlicher Paketraten optimal auszuführen.

Keine Priorisierung stabiler Paketströme möglich

Die alleinigen Ereignisse, die beim reaktiven Server Aktionen auslösen, sind die empfangenen Audiopakete. Jedes einzelne aktiviert den Server – und zwar gleich gewichtet. Der Empfang eines Paketes innerhalb eines höchst regelmäßigen Paketmusters wird genauso bearbeitet wie jedes Paket eines gleichzeitig ankommenden Paketbündels. Intuitiv und naheliegend wäre es dagegen, dem stabilen Client mit regelmäßigem Paketmuster eine höhere Bedeutung beizumessen, nicht zuletzt weil, eine kontinuierliche und regelmäßige Abarbeitung der Aufgaben die Stabilität des Servers erhöht.

Um eine solche Priorisierung vornehmen zu können, muss der Server um Mechanismen erweitert werden, mit denen er die Güte eines Clients bewerten kann.

Paketverluste können nicht sinnvoll zur Kompensation verwertet werden

Ein zentrales Kriterium unseres Entwurfs des Synchronisationsalgorithmus ist die Kopplung von Kompensationsmaßnahmen und Paketverlusten. Wir wollen durch Paketverluste entstandene Lücken in

den Datenströmen bei der Adaption des relativen Versatzes zueinander so nutzen, dass wir Stauchungen oder Dehnungen genau an diesen Stellen vornehmen, um den wahrgenommenen Qualitätsverlust des Audiosignals zu minimieren. Auch für diese Funktion muss unser Basisverfahren erweitert werden.

Über diese Auflistung von Verbesserungsmöglichkeiten unseres Verfahrens bilden sich im Wesentlichen zwei Kenngrößen heraus, die für deren Umsetzung erforderlich sind:

1. die Paketrate jedes Clients
2. die Güte des Paketmusters jedes Clients

Beide Größen sind im Server ohne zusätzliche Informationen allein aufgrund der Paketempfangsmusters ermittelbar, wie in den folgenden Abschnitten beschrieben wird.

6.7.6.2 Bestimmung Client-individueller Paketraten

Aus der nominellen Pufferlatenz von $t_\varphi = 2.66\,ms$ folgt unmittelbar die nominelle Paketrate von $p_{r(nom)} = 375\,Hz$. Wir wissen aus Abschnitt 6.6, dass die Paketraten jedes Clients aufgrund von Ungenauigkeiten der verwendeten Taktgeber von dieser nominellen Rate abweichen, und zwar um einen systematischen Anteil bedingt durch bauliche Toleranzen und einen variablen Anteil, der sich aus variierenden Betriebsparametern ergibt.

Die Paketrate eines Clients kann vom Server abgeschätzt werden anhand der Anzahl der Pakete, die innerhalb einer definierten Zeit von ihm empfangen werden. Äquivalent und für unsere Zwecke besser geeignet ist die Berechnung der Paketrate anhand der Zeit, die für den Empfang einer definierten Anzahl von Paketen gemessen wird. Zwar haben wir festgestellt, dass sich die Uhr des Servers nicht zur absoluten Zeitmessung eignet, für relative Messungen der Clients zueinander gilt diese Einschränkung dagegen nicht. Gleiches gilt für die Variabilität des Zeitgebers beim Server: da Messungen kontinuierlich für alle Clients vorgenommen werden, fließt ein driftender Taktgeber in allen zu gleichen Anteil ein und beeinträchtigt Abschätzungen über relative Zeitmessungen nicht.

Die Abschätzung der Paketrate erfolgt anhand eines Wertepaares M_k, das jeweils die Sequenznummer $seq(p_k)$ und den zugehörigen Empfangszeitpunkt $t(p_k)$ eines Paketes enthält, $M_k = \{seq(p_k), t(p_k)\}$. Die Paketrate entspricht dabei dem Quotienten aus Paketanzahl pro Zeiteinheit, die sich wiederum direkt aus der Differenz der Wertepaare für zwei Pakete p_k und p_m ergeben, es gilt $p_r(k,m) = \frac{\Delta seq}{\Delta t} = \frac{seq(p_m) - seq(p_k)}{t(p_m) - t(p_k)}$.

Zur Berechnung wird als Referenz die Sequenznummer $seq(p_k)$ und deren Empfangszeitpunkt t_k beim Server vermerkt. Die mittlere Paketrate kann anschließend mit dem Empfang eines Paketes p_{k+i} und dem zugehörigen Empfangszeitpunkt t_{k+i} abgeschätzt werden zu

$$p_r(k, k+i) = \frac{\Delta seq}{\Delta t} = \frac{seq(p_{k+i}) - seq(p_k)}{t_{k+i} - t_k} = \frac{i}{t_{k+i} - t_k}$$

Für die Abschätzung des langfristigen Mittelwerts wählen wir die größtmögliche Zeitspanne für die Berechnung, indem wir als Referenz $M_0 = \{0, t(p_0)\}$ wählen. Die so ermittelbare mittlere Paketrate eines Clients C bezeichnen wir $\bar{p_r}(C)$, die sich mit jedem Empfangszeitpunkt t_k eines Paketes $p_{k(C)}$ berechnet zu $\bar{p_r}(C)(k) = \frac{k}{t_k - t_0}$. Dieser Wert nähert sich monoton mit steigender Sequenznummer der mittleren Paketrate eines Clients seit Anfang der Sitzung an, da Abweichungen einzelner Messungen gemittelt werden und so das Ergebnis stetig weniger beeinflussen.

Zur Erkennung driftender Paketraten muss zusätzlich zum langfristigen Mittelwert ein Momentanwert abgeschätzt werden, den wir als Mittelwert über einen kleineren Zeitraum bestimmen. Die Messungen finden dann in festgelegten Abständen zwischen Δ_{seq} Paketen statt. Um nicht kontinuierlich die zur Berechnung erforderlichen Wertepaare M_i im Server zwischenspeichern zu müssen, werden die Momentanwerte nicht

für jedes Paket berechnet, sondern in Abständen von Δ_{seq}. Zu jedem Paket p_k mit $k \geq \Delta_{seq}$ berechnet sich die momentane Paketrate $p'_r(k)$ zu

$$p'_r(k) = p'_r\left(\left\lfloor \frac{k}{\Delta_{seq}} \right\rfloor \cdot \Delta_{seq}\right)$$

Die Wahl des Mittelungsintervalls Δ_{seq} erfordert dabei eine Abwägung zwischen der erwünschten Auflösung und immanent vorhandener Ungenauigkeit. Vor dem Hintergrund, dass die Variabilität der Paketrate durch eine Änderung der Betriebsparameter (wie Temperaturschwankungen) hervorgerufen wird, und diese wiederum aufgrund physikalischer Gesetze beschränkt sind, verwenden wir in unserem NMP-System Mittelungsintervalle von 20 Sekunden. Das entspricht einer Paketanzahl von $\Delta_{seq} = 375\,Hz \cdot 20\,s = 7500$.

Diese beiden lang- und kurzfristigen Mittelwerten für Paketraten lassen sich allein auf Basis der verfügbaren Empfangszeitpunkte und der Sequenznummern mit vernachlässigbaren Aufwand für jeden Client einer Sitzung berechnen. Während unserer Untersuchungen ist der langfristige Mittelwert $\bar{p_r}$ über die Dauer typischer Sitzungen (etwa zwei Stunden) überwiegend konstant genug gewesen, um die benötigten Abschätzungen treffen zu können, während p'_r bei längeren Sitzungen als zusätzliches Kriterium verwendet wurde.

6.7.6.3 Abschätzung der Güte der Paketübermittlung

Mit der Verfügbarkeit einer zuverlässigen Abschätzung für die Paketrate eines Clients ist es nun nicht mehr schwierig, eine Abschätzung für die Güte der Paketübermittlung vorzunehmen. Wir kennen das ideale Paketankunftsmuster, bei dem die Pakete eines Clients C exakt in Abständen von $\frac{1}{p_{r(C)}} = t_{\varphi(C)}$ beim Server eintreffen. Ein Gütekriterium ist entsprechend der Grad der Übereinstimmung der tatsächlich gemessenen Ankunftszeiten mit diesem Ideal.

Naheliegend wäre hierbei, beginnend von einem Referenzpaket die Messungen von idealer und tatsächlicher Ankunftszeit der folgenden Pakete vorzunehmen und die statistische Varianz über die Abweichungen zu ermitteln. Schwierig und für unseren Einsatz hinderlich ist hierbei die Problematik bei die Wahl eines geeigneten Referenzpaketes. Ein beliebig gewähltes Paket kann gerade mit einem hohen Jitter behaftet sein und so die relativ dazu berechneten Abstände zu den Folgepaketen verfälschen. Zwar kann eine solche ungünstige Wahl im Nachhinein erkannt und korrigiert werden, wofür allerdings die Speicherung aller Messungen und eine aufwändige Berechnung nötig sind.

Dieser Aufwand ist unnötig, denn einen Aufschluss über die Güte der Paketübermittlung kann leicht aus der Varianz der Paketzwischenankunftszeiten abgeleitet werden. Dieser Wert wird in ähnlicher Form beim RTCP [84, Anhang A.8] berechnet und als Teil des Sender Reports versendet. Der Jitter wird anhand der Übertragungszeiten berechnet, die wiederum aus Zeitstempeln für Sende- und Empfangszeitpunkt ermittelt werden. Dabei wird für ein Paket p_i die Übertragungslatenz $l_t(p_i)$ berechnet und die Abweichung im Vergleich zum vorherigen Paket $d = |l_t(p_i) - l_t(p_{i-1})|$ ermittelt. Mit dieser Differenz wird abschließend der gleitende Jitter j nach der Formel

$$j = j + \frac{d - j}{16}$$

aktualisiert. Die aktuelle Abweichung fließt somit nur marginal bei der Berechnung von j ein und entspricht einer Tiefpassfilterung der Werte, womit starke Ausschläge bei den Einzelwerten geglättet werden.

Wir können diesen Ansatz aus mehreren Gründen nicht direkt für NMP anwenden. Das Fehlen eines gemeinsamen Zeitnormals laut Ergebnissen aus Abschnitt 6.6 verhindert die Verwendung von Zeitstempeln zur Ermittlung der Übertragungsdauer. Weiterhin wissen wir, dass die Clients mit unterschiedlichen Paketraten arbeiten, so dass der Jitter für jeden Client mit seiner spezifischen Paketrate $t_{\varphi(C)}$ zu berechnen ist.

Bei Verwendung der Abstände benachbarter Empfangszeitpunkte zur Ermittlung von Abweichungen bei der Übertragungslatenz verlieren wir die Information darüber, wo genau die Varianzen verursacht wurden.

Während bei der Berechnung mit Zeitstempeln die Schwankungen präzise auf den Übertragungsweg zwischen den Netzwerkschnittstellen zurückgeführt werden können, schließen die Abweichungen bei den Paketzwischenankunftszeiten den vollständigen Datenpfad zwischen Audiokarte des Clients und Empfang des Paketes beim Server ein.

Tatsächlich ist diese Abweichung für unsere Betrachtung vorteilhaft, denn wir interessieren uns für die Güte des Paketmusters als Ganzes statt der isolierten Betrachtung des Netzpfades. Im Ergebnis können ein Client mit idealer Netzanbindung bei hohen Verarbeitungsjitter im Rechner und ein Client ohne Verarbeitungsschwankung mit hohem Netzwerkjitter ähnliche Variationen der Paketzwischenankunftszeiten haben – Eigenschaften, die in NMP gleich zu behandeln sind.

Die Abweichung der gemessenen Paketzwischenankunftszeiten eines Clients C von seinem Idealwert $t_{\varphi(C)}$ für zwei benachbarte Pakete p_n und p_{n+1} ergibt sich aus $\Delta_{tC}(n+1) = \left|(t_C(p_{n+1}) - t_C(p_n)) - t_{\varphi(C)}\right|$. Im Idealfall beträgt diese Null, höhere Beträge implizieren dagegen Unregelmäßigkeiten beim Paketempfang. Die Summe der Abweichungen während einer Zeitspanne zwischen den Ankunftszeiten $t_C(p_n)$ und $t_C(p_{n+k})$ bietet sich als Gütekriterium für die Stabilität an, die berechnet wird als

$$G_C(p_n, p_{n+k}) = \frac{\sum_{i=n+1}^{n+k} \left| t_C(p_i) - t_C(p_{i-1}) - t_{\varphi(C)} \right|}{k \cdot t_{\varphi(C)}}$$

G_C gibt uns eine qualitative Abschätzung darüber, wie stark Paketzwischenankunftszeiten im Mittel variieren. Ein Rückschluss darauf, wie hoch der Anteil der für das Mischen verspätet eingegangener Pakete ist, ist damit dagegen nicht möglich. Für diese Aussage bedienen wir uns der in Abschnitt 6.5 vorgestellten Verfahren: zusätzlich zur Berechnung eines Durchschnittswertes für die gemessenen Abweichungen bilden wir diese auf mehreren Klassen ab und zählen die relative Auftrittshäufigkeit innerhalb eines Beobachtungszeitraums. Das resultierende Histogramm wird berechnet zu

$$H_C(b) = \frac{\sum_{i=n+1}^{n+k} h'_C(i)}{k} \qquad \text{mit} \qquad h'_C(i) = \begin{cases} 1 & \text{für } \left\lceil \frac{\Delta_t C(i)}{\varphi(C)} \right\rceil = b \\ 0 & \text{sonst} \end{cases}$$

Das Histogramm erlaubt – ähnlich wie bei der adaptiven Bestimmung der Länge des Abspielpuffers beim Client in Abschnitt 6.5 – eine direkte Abschätzung darüber, wie viele Pakete bei einer gegebenen Länge der Warteschlange für das Mischen verspätet eintreffen. Zusammen mit der qualitativen Abschätzung durch G_C und der ermittelten individuellen Paketrate $t^{-1}_{\varphi(C)}$ können wir das im folgenden Absatz vorgestellte proaktive Verfahren umsetzen.

6.7.6.4 Funktionsweise des proaktiven Ansatzes

Die Erweiterung des vorgestellten passiven Basisverfahrens hin zu einer aktiven und vorauseilenden Variante ist durch minimale Anpassungen zu erreichen, die in diesem Abschnitt beschrieben werden. Die resultierenden Eigenschaften und wie damit die Schwächen des Basisverfahrens behoben werden, werden dabei genauer untersucht.

Die erforderlichen Anpassungen beschränken sich darauf, dass statt einer einheitlichen Länge des Warteschlangen-Arrays die Länge für jeden Client individuell der Güte seiner Datenübertragung angepasst wird. Die übrige Bearbeitung, einschließlich der Funktionsweise der aufgestellten Richtlinien zur Bestimmung des Mischzeitpunktes, bleibt gleich. Durch die individuelle Länge der Warteschlange ändert sich lediglich das Entscheidungskriterium von R2 dahin gehend, dass ein Überlauf für jeden Client in Abhängigkeit dieser Länge zu ermitteln ist. Formal ist anzupassen:

ursprünglich R2/PUSH:

$$R2 = ((num_{p(A)} \geq 4) \vee (num_{p(B)} \geq 4) \vee (num_{p(C)} \geq 4))$$

angepasst R2/PUSH:

$$R2 = ((num_{p(A)} \geq l_Q(A)) \vee (num_{p(B)} \geq l_Q(B)) \vee (num_{p(C)} \geq l_Q(C)))$$

Die Werte mit $l_Q(A)$, $l_Q(B)$ und $l_Q(C)$ entsprechen dabei der anhand der Güteparameter ermittelten Länge des De-Jitter Puffers für die Clients A, B und C. Für deren Ermittlung wird kontinuierlich das Histogramm des Jitters der Paketzwischenankunftszeiten $H_C(b)$ berechnet. Analog zu Abschnitt 6.5 kann anschließend am Histogramm direkt abgelesen werden, wie viele Pakete von einem Client vorzuhalten sind, um beim zuletzt gemessenen Paketankunftsmuster innerhalb einer gegebenen Toleranzschranke e_{tol} für Paketverluste zu bleiben. Formal berechnet sich dann die individuelle Pufferlänge eines Clients C zu

$$l_Q(C) = b_e : \sum_{i=0}^{b_e-1} H_C(i) \leq e_{tol} < \sum_{i=0}^{b_e} H_C(i)$$

Nehmen wir zur Veranschaulichung beispielhaft folgendes Szenario an:

- Client A hat ideales Verarbeitungsverhalten und ist im LAN mit dem Server verbunden. Seine Paketzwischenankunftszeiten variieren kaum und führen zu einem gemessenen Histogramm $H_A = \{0.982, 0.015, 0.003\}$
- Client B ist ein entfernter Rechner, die Netzdistanz über mehrere Hops führen zu einem höheren Jitter und einem ermittelten Histogramm von
  $H_B = \{0.58, 0.315, 0.068, 0.015, 0.007\}$. Die Summe aller Einzelwerte von 0.985 sagt aus, dass in der letzten Messperiode 1.5 % Paketverluste verzeichnet wurden.
- Client C ist ebenfalls im LAN mit dem Server verbunden, allerdings sorgt ein schlecht implementierter Gerätetreiber dafür, dass die ISR der Audiohardware regelmäßig ausgebremst wird und die Verarbeitung periodisch gestaut wird. Das resultierende Histogramm $H_C = \{0.97, 0.005, 0, 0.02, 0.005\}$ erlaubt einen unmittelbaren Rückschluss auf die ursächliche Störung.

Für verschiedene Toleranzwerte e_{tol} ergeben sich damit folgende Werte für die Länge der De-Jitter Puffer:

Pufferlänge	Toleranzwerte e_{tol}		
	0.01	0.02	0.04
$l_Q(A)$	2	1	1
$l_Q(B)$	–	5	3
$l_Q(C)$	4	4	1

Auch hier zeigt sich wiederum, dass eine starre Vorgabe für die Toleranz zu unvorteilhaften Ergebnissen führen kann. Deutlich wird das für die Berechnung von $l_Q(C)$, bei der die erforderliche Länge des Puffers zwischen den Toleranzwerten 0.02 und 0.04 sprunghaft von vier auf eins fällt. Das beheben wir mit einer Bewertungsfunktion analog zu der in Abschnitt 6.5 vorgestellten, die einem Tupel aus Toleranz und Latenz einem Zufriedenheitswert zuordnet. Diese Matrix kann bei Bedarf vom Benutzer angepasst werden, für den praxistauglichen Einsatz sind in NMP Standardmatrizen vorhanden. Die Suche nach einem lokalen Optimum erfolgt iterativ nach dem dort vorgestellten Verfahren.

Die Häufigkeit, mit der eine adaptive Anpassung der De-Jitter Puffer vorgenommen wird, ist beim Server nicht beschränkt, da Änderungen der Pufferlängen – anders als beim Client – nicht unmittelbar zu Diskontinuitäten im Audiodatenstrom führen. Effektiv kommen bei NMP Adaptionen meist während einer kurzen Anlaufphase nach dem Start einer Sitzung vor. Innerhalb dieser Initialisierungsphase von der Dauer weniger Sekunden wird l_Q für alle Clients bestimmt, die anschließend über längere Zeiträume konstant bleiben.

Wie diese adaptive Dimensionierung der De-Jitter Puffer hilft, die oben aufgelisteten Schwächen des Basisverfahrens zu beheben, soll hier kurz erläutert werden:

Abhängigkeit der Pufferstände vom Beitrittszeitpunkt zur Sitzung

Wenn ein Client mit geringem Jitter eine Sitzung eröffnet, führt das immer noch dazu, dass seine Warteschlange sich als erstes füllt und ohne weitere Ereignisse gefüllt bleibt. Nachdem die Konstanz seiner Datenübertragung detektiert ist und die Adaption zu einer Verkleinerung dieser führt, reduziert sich die Verweildauer seiner Daten auf die für die Kompensation seines Jitters tatsächlich benötigte Dauer.

Statische Dimensionierung der Warteschlange führt zu Worst Case Latenz

Das im Basisverfahren immanent vorhandene kontinuierliche Anwachsen der Füllstände der De-Jitter Puffer bleibt erhalten. Allerdings hat jetzt jede Queue eine optimale Länge, womit die Daten jedes Clients nur so lange vorgehalten werden, wie für die Verarbeitung seines Paketmusters nötig sind.

Anpassung von Paketratenunterschieden führt meist zu partiellem Mischen

Damit der Datenverwurf von Clients mit einer höheren Paketrate nicht in mehreren partiellen Mischvorgängen mündet, ist eine Berücksichtigung der individuellen Paketrate nötig. Führt der Überlauf eine Warteschlange zur Anwendung der PUSH Regel, wird zunächst geprüft, ob der betreffende Client die höchste Paketrate hat. In diesem Fall kann ein Versatzanpassung direkt durch den Verwurf eines Paketes dieses Clients erfolgen.

Keine Priorisierung stabiler Paketströme möglich

Die Adaption der Warteschlangen bringt eine Priorisierung von Clients mit geringer Schwankung der Paketzwischenankunftszeiten mit sich. Im Idealfall benötigt ein Client genau einen Puffer, wenn die Paketabstände äquidistant sind. Dann fungiert dieser Client als Taktgeber und bestimmt über die PUSH Regel die Mischzeitpunkte. Ein Client mit einem hohen Jitter hat dagegen eine lange Warteschlange und dadurch selten Gelegenheit, das Verfahren zu beeinflussen.

Paketverluste können nicht sinnvoll zur Kompensation verwertet werden

Wird ein Paketverlust in der Warteschlange eines Clients detektiert, können mithilfe der errechneten Client individuellen Paketraten ohnehin erforderliche Korrekturen vorauseilend vorgenommen werden, um die Audioqualität zu optimieren. Ein fehlendes Paket bei einem Client mit relativ hoher Paketrate kann kompensiert werden, indem der anstehende Paketverwurf genau an dieser Stelle durchgeführt wird. Trifft der Paketverlust einen Client mit relativ geringer Paketrate, kann die Lücke zur Dehnung des Datenstroms verwendet werden.

Diese theoretischen Überlegungen haben wir praktisch nachgewiesen. Ohne auf einzelne Messungen explizit einzugehen, sind folgende Eigenschaften herauszustellen:

1. Bei einer Mittelungszeit über 1500 Pakete bzw. vier Sekunden sind die ermittelten Statistiken und Histogramme zuverlässig stabil und ermöglichen nach drei bis vier Durchläufen nach dem Beitritt zur Sitzung die Dimensionierung der De-Jitter Puffer. Diese bleiben meist über lange Zeiten oder gar während der ganzen Sitzung konstant.
2. Clients mit sehr unterschiedlicher Güte der Datenübertragung können miteinander musizieren, ohne dass Clients mit geringem Jitter durch die Teilnahme derer mit hohem Jitter beeinträchtigt wird.
3. Paketverluste und Kompensation unterschiedlicher Paketraten können nahezu vollständig kombiniert werden und die Anzahl von Diskontinuitäten im Audiodatenstrom signifikant verringern. Unsere Messungen ergaben immer Werte für die Überlappung von mehr als 80 %.

6.7.7 Fazit

Ausgehend von den besonderen Anforderungen an die Synchronisation in einem NMP-System haben wir zunächst ein Basisverfahren für die Synchronisierung multipler Audiodatenströme entworfen. Dieses zeichnet

sich durch seine Einfachheit bei gleichzeitig hoher Effizienz und Vorhersagbarkeit aus. Es ist reaktiv, kommt ohne eigene Zeitbasis aus und verwendet lediglich zwei Richtlinien, um die Synchronisation zu gewährleisten und die Mischzeitpunkte zu ermitteln.

Wir haben gezeigt, wie dieses Verfahren in der Lage ist, eine maximale Verweildauer der Audiodaten im Server zu gewährleisten und gleichzeitig allen ermittelten Anforderungen genügt. Anhand von Simulationen wurde das Verhalten der De-Jitter Puffer Queues in allen zu bewältigenden Situationen diskutiert. Die Korrektheit bei der Verarbeitung von Paketstauungen und -schüben, Paketverlusten und unterschiedlichen Paketraten wurde nachgewiesen.

Für Sitzungen, in denen die Abspielverzögerung beim Client dynamisch an das empfangene Paketmuster angepasst wird, offenbart das Basisverfahren Schwächen, da es zu hohen Füllständen der Puffer Queues tendiert und nicht zwischen Paketmustern unterschiedlicher Güte differenziert. Ein Erweiterung des Basisverfahrens hin zu einer proaktiven Variante wurde anschließend abgeleitet, die alle ermittelten Schwächen behebt und darüber hinaus mit einer Kombination der Kompensationsmaßnahmen die Anzahl wahrnehmbarer Störungen minimiert.

Am Schluss ist zu erwähnen, dass die Verfahren unterschiedliche Aspekte von Fairness umsetzten: während das Basisverfahren alle Clients insoweit fair behandelt, dass für jeden die gleiche maximale Verweildauer seiner Daten im Server eingehalten wird, richtet sich die Fairness beim proaktiven Verfahren an die einzelnen Clients: diejenigen mit geringem Jitter werden mit einer geringen Verweildauer belohnt. Auf eine reale Bühne abgebildet entspricht das einem Aufbau, bei dem einige Musiker nahe beieinander sind und andere einen größeren Abstand zur Gruppe haben.

Welche Fairness der Musiker vorzieht, kann bei NMP gewählt werden, indem eine statische oder adaptive Systemlatenz gewählt wird. Da die Wahl einer adaptiven Synchronisation beim Server nur bei einer adaptiven Anpassung des Abspielpuffers beim Client sinnvoll ist und im umgekehrten Fall gar nur dann funktioniert, betrifft die Wahl zwischen statischem oder adaptiven Verhalten immer beide Komponenten.

KAPITEL 7

DESIGN

Aufgrund der in Kapitel 3 aufgeführten Herausforderungen ist die Realisierung eines NMP-Systems eine komplexe Aufgabe. Das im Fokus dieser Arbeit behandelte Echtzeitszenario erfordert die isochrone Übertragung und Verarbeitung von Audiodaten bei geringer Latenztoleranz. Gleichzeitig müssen für eine sinnvolle Verwendung des Systems auf diese Basiskomponente aufsetzende Funktionen und Dienste verfügbar sein, die den genannten Einschränkungen zuwiderlaufen.

In diesem und dem folgenden Kapitel wird beschrieben, wie wir das Dilemma durch eine Umsetzung des Systems als Kombination von lose miteinander gekoppelten Komponenten umgesetzt haben. Gedacht sind die Ausführungen für diejenigen, die unser NMP-System als Schablone für ähnliche Anwendungen verwenden wollen oder an der praktischen Umsetzung der beschriebenen Herausforderungen interessiert sind. Allen anderen Lesern wird empfohlen, diesen praktischen Teil zu überspringen und direkt zu den Evaluationsergebnissen in Kapitel 9 zu springen.

Als Ergebnis unserer Latenzanalyse in Kapitel 5 haben wir ermittelt, dass den grundlegenden Gegensätzen mit einer Aufteilung der Funktionseinheiten in Echtzeitkomponenten für die isochrone Verarbeitung und allen übrigen Funktionseinheiten begegnet werden kann. Wir haben in Abschnitt 5.6 dargestellt, wie eine Umsetzung von Echtzeitclient und -Server erfolgen muss, um die ermittelten Latenzwerte praktisch umsetzen zu können. Das dort beschriebene systemnahe Vorgehen erfordert unterschiedliche Ansätze für den Entwurf beider Komponenten: bei den Echtzeitkomponenten bestimmt allein die Latenz unter Einhaltung der in Kapitel 6 diskutierten Lösungsansätze das Design, während der Entwurf der übrigen Komponenten keinen spezifischen Einschränkungen unterliegt, da diese lediglich die im Echtzeit-Betrieb gesammelten Daten über wohlbekannte Methoden weiter verarbeiten.

Der Entwurf sieht vor, dass die Echtzeitkomponenten ein autarkes System bilden, das über definierte Schnittstellen von den darüber liegenden Komponenten über ein Signalisierungsprotokoll kontrolliert und über dieses mit Daten für die nachträgliche Bearbeitung versorgt wird. Die Trennung von Signalisierung und Nutzdatentransport ist eine verbreitete und bewährte Methode, wenn eine kontinuierliche und latenzkritische Datenübertragung durch eine darüber liegende komplexere Anwendung kontrolliert wird. Hier seien exemplarisch das Zusammenspiel von RTSP und RTP bzw. der Verbindungsauf- und Abbau und der eigentliche Datentransfer in P2P-Netzen genannt.

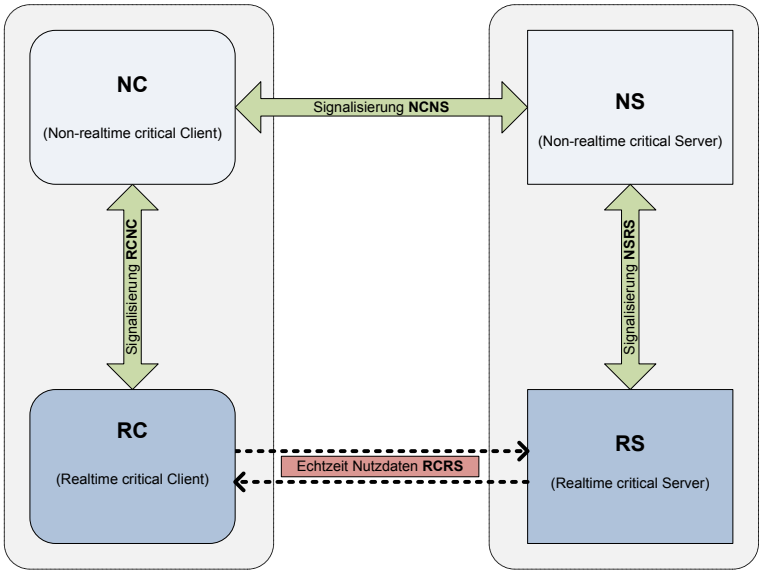

Abbildung 7.1: Modularer Aufbau der NMP Anwendung aus Echtzeit- und zeitunkritischen Komponenten

Im Folgenden beschreiben wir den Entwurf unseres NMP-Systems. Wir beginnen mit einem Grobkonzept des angestrebten modularen Designs und leiten aus einer Diskussion die Anforderungen und Einschränkungen an die Architektur ab, die unser System als verteilte Anwendung von miteinander kommunizierenden Komponenten definiert. Wir blicken anschließend genauer in jede dieser Komponenten und gehen auf für das Design relevante Besonderheiten ein. Die Protokolle für die Kommunikation zwischen den Komponenten werden separat betrachtet.

7.1 Grobkonzept

Die prinzipielle Umsetzung unseres NMP-Systems geht direkt aus den bisher durchgeführten Untersuchungen und den sich daraus ergebenden Anforderungen hervor. Wir haben bei der prototypischen Umsetzung der Endsysteme unter Einhaltung der analytisch bestimmten Latenzschranken ermittelt, dass sowohl Client als auch Server modularisiert werden müssen, damit die Echtzeitverarbeitung nicht von der Signalisierung behindert werden. Diese Forderung impliziert, dass mindestens je eine Komponente beim Client und Server vorhanden sind, die alle in Kapitel 6 gewonnenen Ergebnisse umsetzen und dabei eine Systemlatenz innerhalb der Toleranzgrenzen gewährleisten.

Die nach diesem Kriterium unterscheidbaren Teile werden im Weiteren auf der Clientseite *RC* (für Realtime Client) und *NC* (für Non-realtime Client) bezeichnet. Analog befinden sich auf der Serverseite die Komponenten *RS* und *NS*. Abbildung 7.1 skizziert diese Modularisierung schematisch. In Anlehnung an existierende Schichtenmodelle nehmen wir hiernach für NMP eine Aufteilung in zwei Schichten an: RC und RS liegen in der untere bzw. Transportschicht, NC und NS in der oberen bzw. Signalisierungsschicht.

Die Grundfunktionalität unseres NMP-Systems erfüllen dabei RC und RS, die – einmal angestoßen – autonom die isochrone Verarbeitung der Audiodaten erledigen. Dem RC fällt dabei die Aufgabe zu, Audiopakete von der Soundkarte abzugreifen und schnellstmöglich zum RS zu versenden, sowie auf der Rückrichtung die gemischten Audiopakete zu empfangen und auszugeben. Die Kommunikation zwischen diesen beiden Modulen findet über das Echtzeitprotokoll *RCRS* statt.

Alle auf diese Basisfunktionalität aufbauenden Features werden von den darüber liegenden zeitunkriti-

schen Komponenten NC und NS zur Verfügung gestellt. Hierzu fallen zunächst die Benutzerschnittstelle und die Signalisierung von RC und RS an, um eine Sitzung anzustoßen. Darüber hinaus kann die Funktionsvielfalt beliebig ausgedehnt werden, um beispielsweise die Verwaltung von Benutzern, Gruppen oder Rechten bereitzustellen. Auch das Aufzeichnen der Musikdaten und die nachfolgende Bereitstellung erfolgen in diesen Modulen. Zwar handelt es sich auch dabei um Echtzeitdaten, jedoch ermöglicht das Fehlen von Interaktivität eine großzügige Dimensionierung von Puffern und eine Verarbeitung der Daten außerhalb der zeitkritischen Routinen.

Für die Kommunikation zwischen den Komponenten eines Endsystems sind die Protokolle *NCRC* beim Client und *NSRS* beim Server vorgesehen. Diese stellen primär Mechanismen für die Kontrolle der zeitkritischen durch die drüber liegenden Komponenten bereit. Darüber hinaus werden damit Statusinformationen und zur Archivierung vorgesehene Daten aus der unteren Schicht an die oberen übermittelt, da wegen der in Abschnitt 5.6 dargestellten Schwierigkeiten Audiodaten nicht in den zeitkritischen Modulen selbst archiviert werden können.

Für die Kommunikation zwischen NC und NS ist das Signalisierungsprotokoll *NCNS* vorgesehen, welches den Zugang des Benutzers zum NMP-System bereitstellt. Die genannten Komponenten und Protokolle werden im Folgenden etwas genauer durchleuchtet.

7.2 Software Architektur

Die Software Architekturen der beiden Endsysteme Client und Server unterscheiden sich aufgrund der bekannten funktionalen Anforderungen signifikant und werden in diesem Abschnitt separat vorgestellt. Allgemeine und von beiden Systemen gemeinsam verwendeten Module werden einleitend in einem eigenen Absatz beschrieben.

7.2.1 Ausgewählte Datenelemente und Funktionseinheiten

Datenklassen, die in unserem NMP-System durchweg und von allen Komponenten verwendet werden, haben wir generalisiert entworfen und als Bibliotheken realisiert. Zu diesen gehören zentrale Element wie das Audiopaket, in dem jeglicher Datenaustausch zwischen Client und Server und die Archivierung der Audiodaten erfolgen. Ebenso sind in der Bibliothek Klassen enthalten, die eine Abstraktion der Hardware und des Betriebssystems erlauben und die Umsetzung plattformunabhängiger Endsysteme erlauben.

7.2.1.1 Datenklassen für Audiodaten

Für die Verwaltung und Bearbeitung von Audiodaten auf den gesamten Datenpfad wird ein Satz von Datenstrukturen bereitgestellt, von denen die wichtigsten in Abbildung 7.2 dargestellt sind.

Die häufigst verwendete ist die `AudioPacket` Datenklasse, in denen die Audiodaten nahezu ihren gesamten Lebenszyklus verbringen. Ein `AudioPacket` Objekt verwendet wiederum zwei grundlegende Klassen, um das Audioformat zu beschreiben und die Daten zu verwalten. Ersteres geschieht über `AudioFormat`, welche die Eigenschaften der im Audiopaket enthaltenen Daten beschreibt. Darunter fallen das für eine Sitzung gewählte Audiodatenformat, welches über den Verlauf einer Sitzung statisch ist, ebenso wie variante Werte wie das verwendete Kompressionsverfahren oder die Position eines Musikers im virtuellen Raum.

Für die Verwaltung und den Austausch von generischen Datenpuffern ist die Klasse `DataBuffer` vorgesehen. Wo immer in NMP Daten in Form von Blöcken oder Strömen verwendet bzw. zwischen Objekten ausgetauscht werden, erfolgt dies über Instanzen dieser Klasse. Für eine effiziente Verwendung ist `DataBuffer`

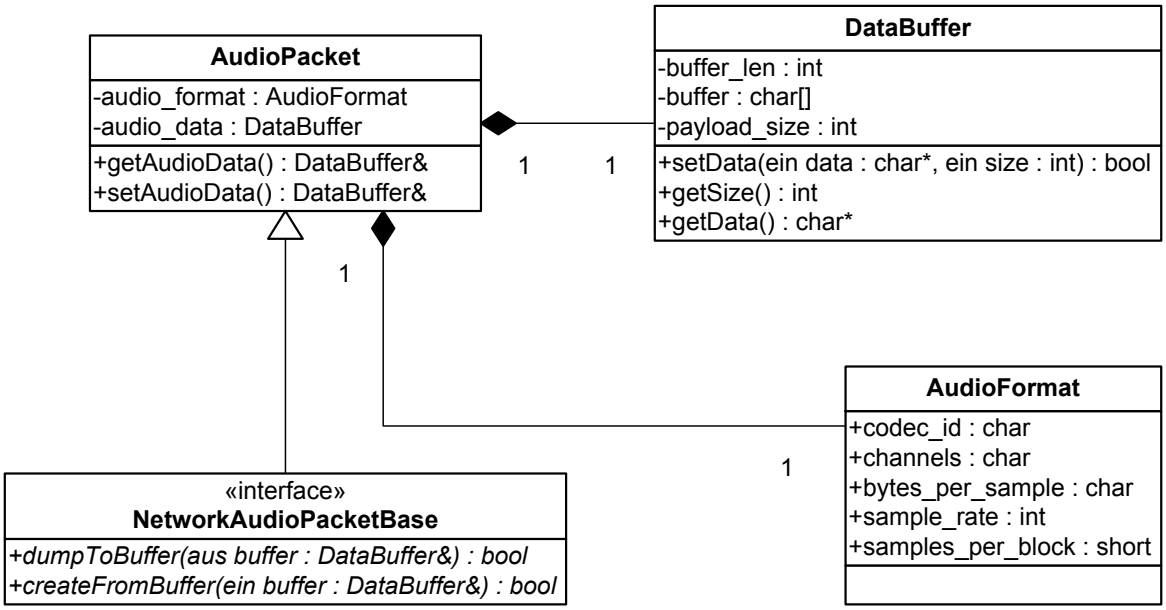

Abbildung 7.2: Zentrale Datenklassen für Audiodaten

so entworfen, dass Daten sowohl als Referenzen auf extern allozierte Speicherbereiche (*shallow copy*) als auch als Kopien der Daten im Instanzspeicher (*deep copy*) verwaltet werden können.

Die Audiodaten werden bei der Erzeugung am Client in einem `DataBuffer` abgelegt und anschließend einem `AudioPacket` Objekt über `setAudioData()` zugewiesen. Alle weiteren Bearbeitungsschritte wie Audiocodierung und Manipulation durchleben die Audiodaten in den Endsystemen als `AudioPacket` Objekte, bis sie schließlich am Ende ihres Weges über `getAudioData()` ausgelesen und über die Audiohardware abgespielt werden.

Für den Transport im Netz müssen die Daten transformiert werden. Diese Operation beinhaltet u.A. die Anpassung der Datenrepräsentation zwischen Endsystem und Netz, die abhängig von der Prozessorarchitektur und/oder dem Betriebssystem eine Umstellung der Byte-Reihenfolge (*Endianess*) beinhalten kann. Die Kapselung dieser Systemabhängigkeit erfolgt über das `NetworkAudioPacketBase` Interface. Dieses erweitert `AudioPacket` um die beiden Methoden `dumpToBuffer()` und `createFromBuffer()`, mit denen der Inhalt eines Audiopaketes in ein `DataBuffer` in Netz Byte-Reihenfolge geschrieben bzw. von dort erstellt werden kann. Von der Basisklasse abgeleitete Spezialisierungen unterstützen die Verwendung unterschiedlicher Paketheader in Senderichtung und deren Differenzierung in Empfangsrichtung, um die Audiodaten dynamisch mit ergänzenden Informationen zu versehen.

7.2.1.2 Betriebssystem Abstraktionslayer `OSALBase`

Eine wichtige selbst gestellte Anforderung an unser NMP-System ist eine weitläufige Plattformunabhängigkeit, um der Diversität eingesetzter Computer- und Betriebssysteme besonders im akademischen und immer häufiger auch im privaten Bereich nachzukommen. Auch eine Umsetzung einzelner Komponenten auf dedizierte eingebettete Systeme, wie wir sie gegen Ende dieses Abschnitts vorstellen werden, erfordert eine Minimierung plattformspezifischer Funktionalität. Diese wird in unserem Entwurf über die in Abbildung 7.3 dargestellten `OSALBase` Layer umgesetzt.

Der Abstraktionslayer für das Betriebssystem (*OSAL: Operating System Abstraction Layer*) bedient sich mehrerer untergeordneter Schnittstellen zu thematisch gruppierten Funktionseinheiten. Für die auf Zeit-

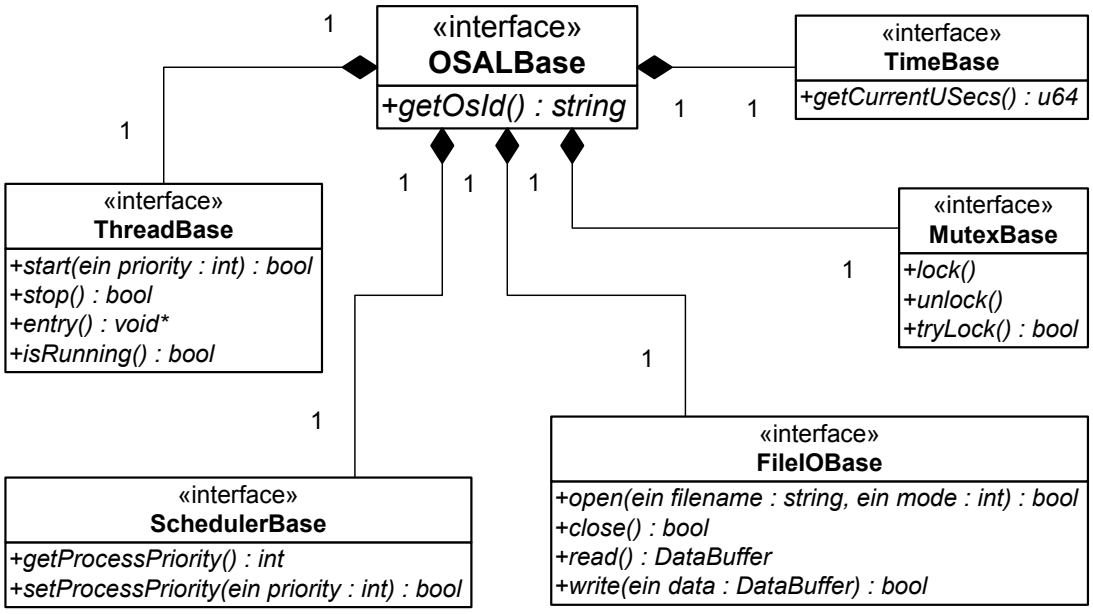

Abbildung 7.3: Kapselung Betriebssystem spezifischer Abhängigkeiten im OS Abstraction Layer (OSAL)

messung basierende Berechnung von Statistiken stellt `TimeBase` die Methode `getCurrentUSecs()` bereit, die die aktuelle Systemzeit mit Mikrosekundengenauigkeit zurückgibt. Da die Endsysteme in NMP ohne gemeinsame Zeitdomäne auskommen, ist für die erforderliche Messung von Zeitdifferenzen von der Systemzeit lediglich eine Kontinuität erforderlich, während der Versatz nicht relevant ist. Damit kann sie bspw. als auf Nanosekunden erweiterte Unixzeit oder als Zeit seit dem letzten Systemstart interpretiert werden.

Die `SchedulerBase` Schnittstelle stellt `getProcessPriority()` und `setProcessPriority()` Methoden bereit, mit denen die Priorität des aktuellen Prozesses abgefragt und bei Bedarf geändert werden kann. Eine Anpassung dieses Parameters ist zumindest für den RS erforderlich, der mit der höchstmöglichen Priorität ausgeführt werden muss, um möglichst verzögerungsfrei auf empfangene Pakete reagieren zu können.

Die hier weiterhin aufgeführten Schnittstellen `ThreadBase`, `MutexBase` und `FileIOBase` sind für die Ausführung von Threads und die Synchronisation zwischen ihnen und für den Dateizugriff erforderlich. Sie werden ausschließlich von den zeitunkritischen Komponenten verwendet, da RS und RC weder Threads verwenden noch auf langsame Peripherie zugreifen dürfen.

7.2.1.3 Zugriff auf Audiohardware über `AudioIoBase`

Die zuverlässige Funktion des RC steht und fällt mit der Möglichkeit, seine Ausführung innerhalb der ISR der Audiohardware zu gewährleisten. Da diese Forderung bereits sehr systemnah ist und wir weiterhin festgestellt haben, dass dafür eine Auswahl unter den gegenwärtigen Audioschnittstellen erforderlich ist, müssen wir bereits beim Entwurf der Schnittstelle zur Audiohardware `AudioIoBase` Einschränkungen hinnehmen und Spezialisierungen vornehmen.

Der abstrakte Zugriff auf die Audiohardware über das `AudioIoBase` Interface zusammen mit den realisierten Spezialisierungen ist in Abbildung 7.4 skizziert. Die Minimalfunktionalität eines AudioIo Objektes stellt eine CallBack Methode `audioIsrCB()` bereit, die im Treiber der Soundkarte registriert wird. Bei jedem Aufruf erwartet der Treiber die Übergabe des nächsten auszugebenden Audioblocks, der über die Methode `putAudioPacket()` zu erfolgen hat. Der zeitgleich zum Auslesen verfügbare Audioblock wird mit `getAudioPacket()` ausgelesen und im Rückgabewert als `AudioPacket` bereitgestellt.

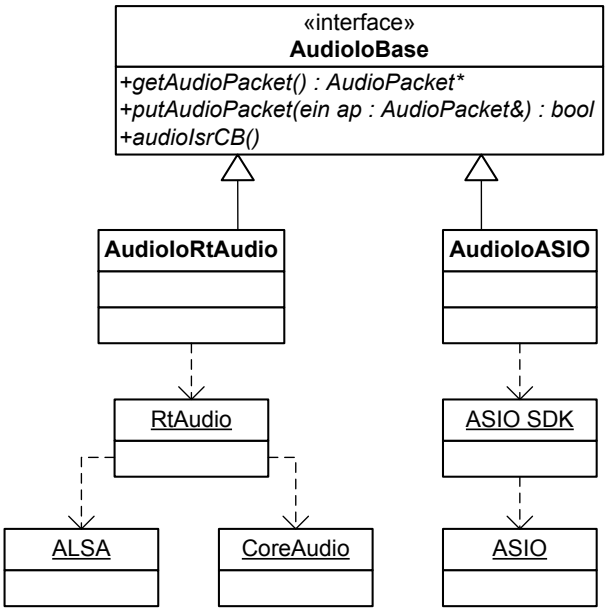

Abbildung 7.4: Schnittstelle zur Abstraktion der Audiohardware

Diese Schnittstelle stellt über Spezialisierungen den Zugriff auf die Audiohardware in den heute gängigsten Systemen Windows, Linux und OS/X bereit. Für eine Minimierung der Plattformunabhängigkeit haben wir zunächst vollständig auf das explizit für solche zeitkritischen Zugriffe entwickelte *RtAudio* Framework [82] aufgesetzt. Diese erlaubt den transparenten Zugriff auf unterschiedliche Audio-Interfaces auf allen genannten Systemen bei minimaler Latenz. Tatsächlich genügt RtAudio unseren strikten Latenzanforderungen beim Einsatz unter Linux und OS/X, zeigt sich auf Windows-Systemen jedoch zu fehleranfällig. Dort greifen wir daher direkt auf das ASIO Interface zu und erreichen damit einen stabilen Betrieb bei minimaler Verzögerung.

Unter diesen Vorgaben wurden von `AudioIoBase` die beiden Spezialisierungen `AudioIoRtAudio` und `AudioIoASIO` entworfen. Die erstere setzt auf RtAudio auf und stellt den Zugriff auf die Audiohardware über die ALSA-Schnittstelle (*Advanced Linux Sound Architecture* [68]) und über Apples CoreAudio bereit. `AudioIoASIO` greift dagegen über Steinbergs ASIO SDK [89] direkt auf den ASIO-Treiber der verwendeten Hardware zu.

7.2.1.4 Netzzugriff über `SocketBase`

Für die Kommunikation im Netz hat sich die *socket*-Schnittstelle etabliert, die in Form der *BSD sockets* integraler Bestandteil aller heute verwendeten Unix Derivate ist. Microsoft bildet diese Schnittstelle mit der *WinSock*-API im Windows-Betriebssystemen weitestgehend vollständig ab, verhindert jedoch mit proprietären Erweiterungen plattformübergreifende Quellcodekompatibilität.

Diese Abhängigkeit wird innerhalb unserer `SocketBase` Klasse gekapselt, deren Schnittstelle in Abbildung 7.5 skizziert ist.

Die `SocketBase` Basisklasse setzt alle System spezifischen Zugriffe auf die jeweilige Socket-API um und bietet den davon abgeleiteten Klassen einen transparenten Zugang zur Socket basierten Kommunikation. Als Spezialisierungen sind die Klassen `UdpSocket` für den verbindungslosen Transport der Audiodaten und `TcpSocket` für den verbindungsorientierten Austausch von Nachrichten vorgesehen.

Ein generisches `TcpSocket` Objekt ist dabei als aktive bzw. Client-seitige Komponente ausgelegt, die

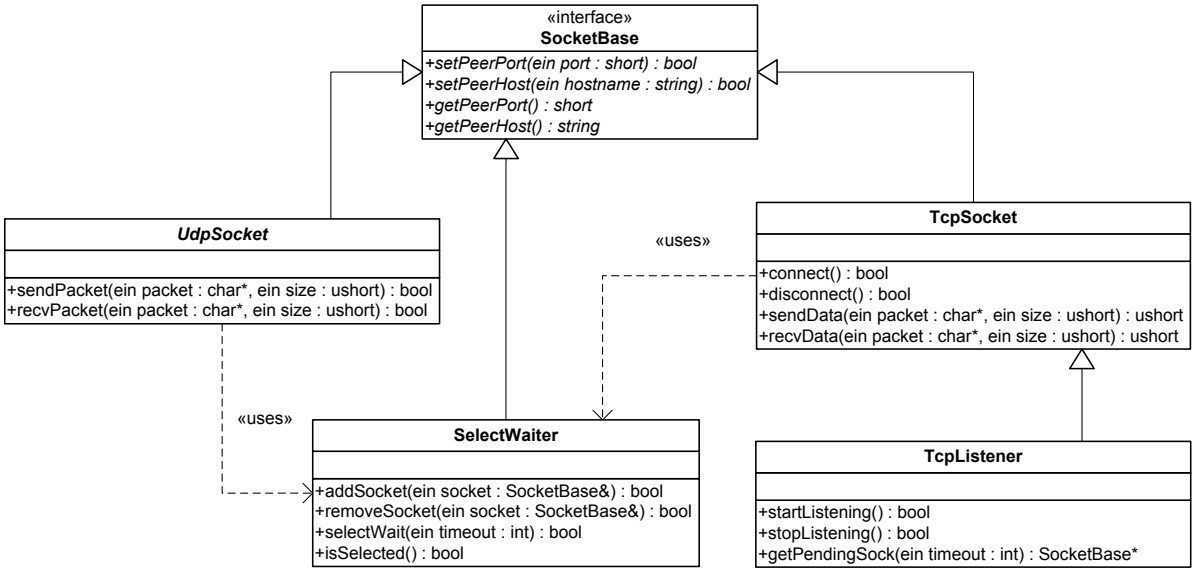

Abbildung 7.5: Klassendiagramm der SocketBase Schnittstelle

eine Verbindung zum Server über die Methoden `connect()` und `disconnect()` initiieren bzw. terminieren kann. Auf der Gegenseite bearbeitet ein `TcpListener` Objekt diese Verbindungsanfragen und stellt über `getPendingSock()` für jede erfolgte Verbindung ein neues `TcpSocket` Objekt bereit.

Für die Verwaltung der Socketzustände wird die `SelectWaiter` Klasse realisiert. Diese kapselt die in der BSD-Socket API verfügbare *select* Funktionalität und erlaubt die simultane Statusabfrage der zu einem Deskriptorsatz bestehende Sockets. `SocketBase` Objekte werden dabei über die Methode `addSocket()` zum Satz des betreffenden SocketListeners hinzugefügt bzw. mit `removeSocket()` entfernt. Ob mindestens einer der verwalteten Sockets Daten empfangen hat und diese zum Auslesen bereithält, liefert der Rückgabewert von `selectWait()`. Ist diese globale Abfrage positiv, wird anschließend für jeden Socket im Satz über `isSelected()` der individuelle Status abgefragt und eine Bearbeitung der empfangenen Daten initiiert.

7.2.1.5 Schnittstelle zur Audiodatenkompression `AudioCodecBase`

Moderne Verfahren zur Audiodatenkompression sind der in Arbeitsplatzrechnern verwendeten Multi-Threading Arbeitsweise angepasst und stellen meist eine asynchrone Benutzerschnittstelle für die Verarbeitung von Audiodatenströmen bereit. Das dabei angenommene Paradigma geht davon aus, dass ein Produzent den Codec kontinuierlich mit Audiodatenschnippseln versorgt, die dieser nach eigenem Ermessen bearbeitet und einem Konsumenten über einen Rückruf bereitstellt. Dieses Verfahren bedient die in Desktop Betriebssystemen verwendeten Semantik von Audiodatenkompressoren, die als Funktionskomponenten entlang eines virtuellen Datenpfades mit kontinuierlichem Datenfluss angenommen werden.

Eine solche Schnittstelle geben auch die in NMP verwendeten Kompressionsverfahren FLAC [97] und WavPack [23] vor, die für unsere Zwecke jedoch unbrauchbar ist, da wir eine synchrone und blockbasierte Bearbeitung der Audiodaten benötigen. Die dafür vorgesehene Schnittstelle `AudioCodecBase` und die davon abgeleitete Klassenhierarchie sind in Abbildung 7.6 dargestellt.

Für den Einsatz in NMP muss ein beliebiger Audiocodec lediglich zwei Methoden bereitstellen: mit `encodeAudioPacket()` wird das übergebene unkomprimierte Audiopaket codiert und mit `decodeAudioPacket()` wieder dekomprimiert. Als Spezialisierungen dieser Schnittstelle sind die Klassen `AudioCodecWavpack`, `AudioCodecADPCM` und `AudioCodecFLAC` implementiert, welche jeweils die asyn-

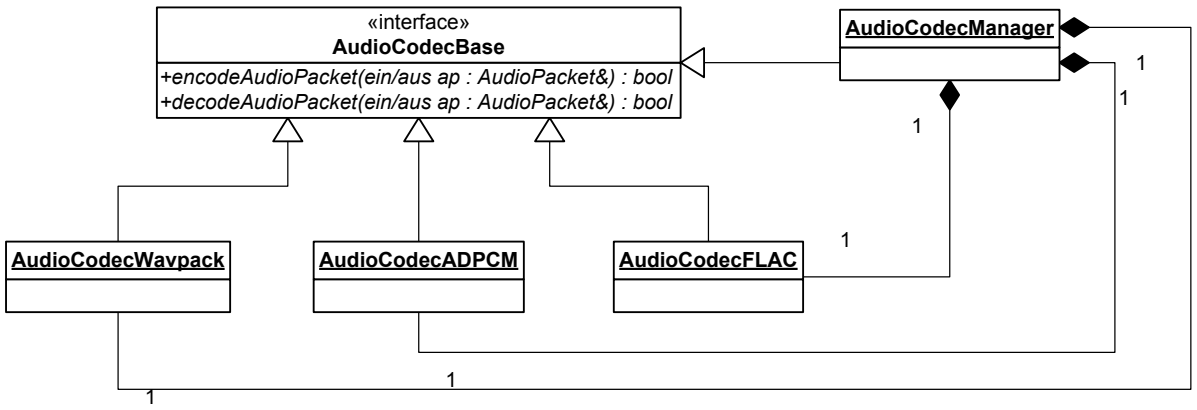

Abbildung 7.6: AudioCodecBase Schnittstelle

chrone und auf Datenstrom basierten Verarbeitung zu einem synchronen und blockbasierten Zugriff kapseln.

Die ebenfalls von `AudioCodecBase` abgeleitete Klasse `AudioCodecManager` stellt einen virtuellen Proxy bereit, der alle in einem System verfügbaren Codecs verwaltet und die Bearbeitung einer Funktion an das korrespondierende Objekt delegiert. Während bei einem direkten Zugriff auf ein `AudioCodec` Objekt die Datenverarbeitung immer ein uncodiertes AudioPaket als Eingabe erwartet bzw. ein solches erzeugt, kann der `AudioCodecManager` unter sequenzieller Abarbeitung des Audiopaketes durch die registrierten `AudioCodec` Objekte ein Audiopaket mit beliebigem Quellformat in ein beliebiges Zielformat konvertieren.

7.2.2 Echtzeitkomponenten

Die beiden Endpunkte auf der Transportschicht unseres Systems bilden der *RS* im Server und der *RC* im Client, die über das verbindungslose Protokoll *RCRS* miteinander verbunden sind. Im Wesentlichen bildet dieser Teil den in dieser Arbeit thematisierten Basisdienst für den latenzarmen Transfer von Audiodaten ab, der die Grundlage für das interaktive Musizieren im Netz ist.

Dieses aus RC, RS und RCRS bestehende Basissystem ist für sich genommen bereits funktional: nach einer Initialisierung und einem Startaufruf wird die Echtzeit-Sitzung ohne weitere äußere Interaktion so lange aufrecht erhalten, bis sie beendet wird.

7.2.2.1 Echtzeit-Client RC

Aufgrund seiner reduzierten Funktionalität lässt sich der RC am anschaulichsten als bidirektionale Netz-Audiokarte idealisieren. Er greift die vom Mikrofon digitalisierten Daten an der lokalen Audiohardware ab, verpackt diese und versendet sie über das Netz an ein vorgegebenes Ziel. Umgekehrt empfängt er von dort Pakete, die er nach dem Auspacken der Audiohardware zuführt und diese dort abgespielt werden.

Die Umsetzung dieses sehr vereinfachten Grundprinzips wird durch die in Kapitel 3 diskutierten Herausforderungen erschwert. Aus den gewonnenen Erkenntnissen leiten sich zahlreiche Entwurfsbedingungen ab.

In Abschnitt 5.6 wurde gezeigt, dass ein Arbeitsplatzrechner aufgrund fehlender Echtzeitfähigkeiten die kontinuierliche Abarbeitung isochroner Daten über den Scheduler nur mit unzureichender Genauigkeit unterstützt. Wir wählen stattdessen die äquidistanten und präzisen Interruptsignale der Audio-Hardware als Aktivierungsereignisse für die Ausführung des RC. Dieser wird als Callback-Hook innerhalb der ISR (Interrupt-

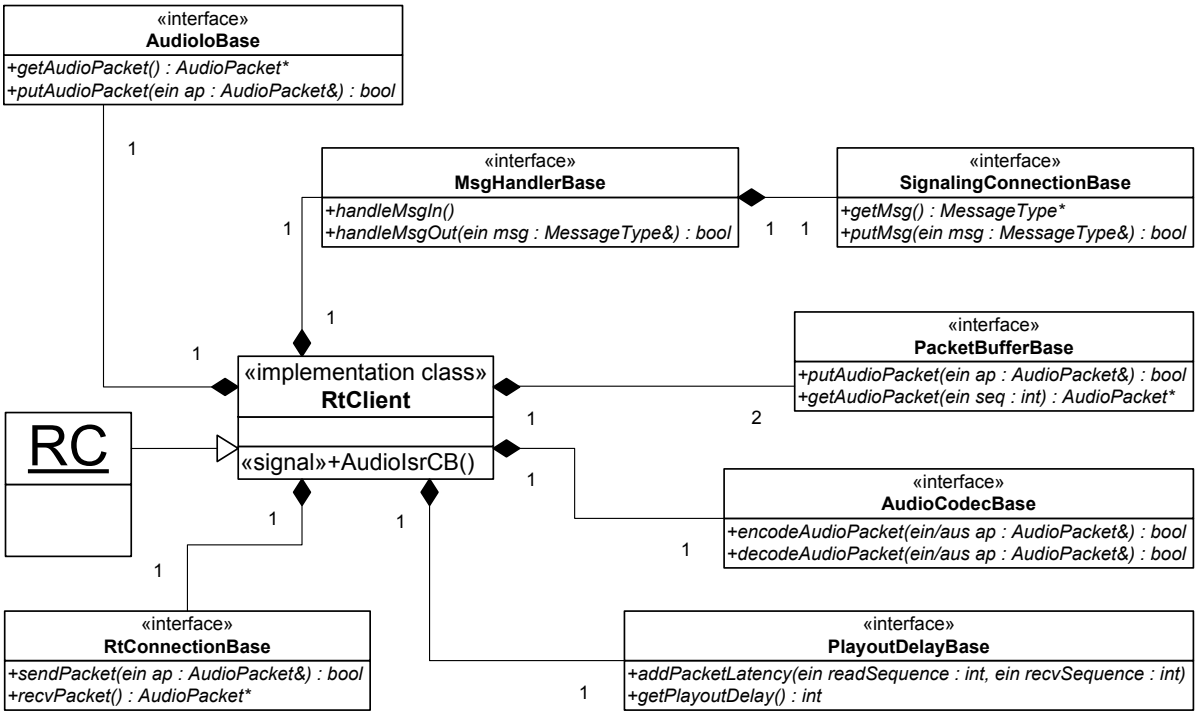

Abbildung 7.7: Klassendiagramm des Echtzeit-Clients RC

Service-Routine) der verwendeten Audio-Schnittstelle realisiert. Dadurch wird gewährleistet, dass er vom Gerätetreiber unmittelbar und direkt aufgerufen wird, sobald die Audiohardware einen Audioblock bereitstellt. Dieses geschieht aufgrund der kontinuierlichen Verarbeitung der Daten mit der gewählten Audiokonfiguration zuverlässig in zeitlich äquidistanten Abständen von t_φ.

Die Ausführung des RC auf einer systemnahen Ebene stellt sicher, dass die Reaktionszeit der Anwendung minimiert wird, da die Verzögerung für das Scheduling im Betriebssystem entfällt. Gleichzeitig wird auch der Anwendungs-Jitter minimiert, da die Callback-Funktion unmittelbar nach Eintreffen des Hardware-Interrupts ausgeführt wird und somit die zeitlichen Abstände zwischen zwei aufeinanderfolgenden Aufrufen konstant sind. Da der Callback im Interrupt-Kontext ausgeführt wird, ist schließlich auch gewährleistet, dass der RC mit sehr hoher Priorität läuft – er kann nur durch einen höher priorisierten Interrupt unterbrochen werden.

Die Umsetzung des RC als Callback-Hook liefert nahezu reale Isochronität bei gleichzeitig einfacher und systemunabhängiger Realisierbarkeit. Es besteht darüber hinaus die Möglichkeit, den RC direkt als Treiber zu realisieren, um auch die verbleibenden Verzögerungen durch den Wechsel vom Kernel- in den Userspace zu entfernen. Angesichts der vernachlässigbaren Größe dieser Verzögerung im Mikrosekundenbereich, die im Gegenzug den Verlust der Plattformunabhängigkeit impliziert, wurde diese letzte Optimierung in dieser Arbeit nicht berücksichtigt.

Die Implementierung des RC im Interrupt Kontext bringt uns zwar nahe an das erforderliche Echtzeitverhalten, erfordert auf der anderen Seite jedoch eine höchst effiziente und eingeschränkte Implementierung des RC. Naheliegend ist zunächst, dass die Aufenthaltsdauer innerhalb der Callback-Routine minimiert werden muss. Sie ist nach oben hin durch t_φ beschränkt, sollte jedoch um Größenordnungen darunter liegen, um den Rechner nicht vollständig auszulasten. Weiter reichend sind Einschränkungen, die den Zugriff auf langsame Ein- und Ausgabegeräte oder den Aufruf von blockierenden Systemroutinen verhindern. So sind jegliche Zugriffe auf Peripherie in Form von Bildschirmausgaben oder Dateizugriffen oder Interaktionen mit

dem Benutzer im RC nicht möglich. Einzig der Zugriff auf die Netzschnittstelle erfolgt relativ verzögerungsfrei, da hierbei die Abarbeitung größtenteils in der Netzhardware erfolgt und sich die Belastung der CPU auf das Kopieren des Paketinhaltes in den Treiberpuffer beschränkt.

Unter Einhaltung dieser Beschränkungen bietet sich die Umsetzung des RC als Daemon an, der einmalig auf jedem NMP-Client ausgeführt und über die Signalisierung nach dem NCRC-Protokoll kontrolliert wird. Neben der Parametrierung des RC beinhaltet dieses Nachrichten zum Anstoßen und Beenden von Sitzungen.

Das Design unseres RC, der den in Abschnitt 5.6 aufgestellten Anforderungen genügt, ist als Klassendiagramm in Abbildung 7.7 dargestellt. Der RC wird dabei als Instanz einer generischen `RtClient` Klasse umgesetzt, die auf das Netz und auf die Audiohardware über definierte Schnittstellen zugreift. Die Realisierung der periodischen Aktivierung innerhalb der Audio-ISR erfolgt über die statische Klassenmethode `AudioIsrCB()`, die bei der Instantiierung des `AudioIO`-Objektes registriert wird. Über diese werden Audioblöcke über `getAudioPacket()` von der Audiohardware gelesen bzw. über `putAudioPacket()` geschrieben.

Für die Anbindung an das Netz sind die Interfaces `RtConnectionBase` für den verbindungslosen zeitkritischen Pakettransport und `SignalingConnectionBase` für den verbindungsorientierten Austausch von Nachrichten für die Signalisierung vorgesehen. Das für die Kommunikation mit dem RS vorgesehene RCRS-Protokoll wird von einem `RtConnection` Objekt realisiert, welches Methoden für das Empfangen und Versenden eines Audiopaketes über `recvPacket()` bzw. `sendPacket()` bereitstellt. Die Signalisierung über NCRC erfolgt dagegen über ein `SignalingConnection` Objekt, das mit `getMsg()` und `putMsg()` über Methoden für Empfang und Versand von Nachrichten verfügt.

Die Verarbeitung der Nachrichten findet aus Modularitätsgründen nicht im `RtServer` selbst statt, sondern ist in der `MsgHandlerBase` gekapselt. Mit `handleMsgOut()` kann der RC eine Nachricht versenden, während ein Aufruf von `handleMsgIn()` eine evtl. im Empfangspuffer vorhandene Nachricht verarbeitet. Das Quittieren verarbeiteter Nachrichten erfolgt innerhalb dieser Klasse ebenso wie die Prüfung der Antwort auf versendete Nachrichten und möglicher Fehlerbehandlungen.

Für die Pufferung von ein- und ausgehenden Audiopaketen ist jeweils ein Objekt der abstrakten Klasse `PacketBufferBase` vorgesehen. Diese verwalten `AudioPacket` Objekte anhand ihrer eindeutigen Sequenznummern, die bei Übergabe per `putAudioPacket()` eingefügt und mit `getAudioPacket()` wahlfrei ausgelesen werden. Der Puffer wird als Ring aufgesetzt, wobei ältere und mit derselben Sequenznummer bereits enthaltene Pakete überschrieben werden. Die in Abschnitt 6.3 beschriebenen Mechanismen für Fehlerverdeckung werden innerhalb dieser Klasse implementiert. Von `PacketBufferBase` abgeleitete Klassen können unterschiedliche Korrekturverfahren enthalten oder ohne diese auskommen.

Über das `AudioCodecBase` Interface werden die in NMP verwendeten Audiokompressionsverfahren angebunden. Die Transformation erfolgt in-place, das Audioformat des übergebenen Audiopaketes wechselt entweder von oder zu PCM.

Zuletzt ist zur Umsetzung des in Abschnitt 6.5 vorgestellten Ansatzes zur Bestimmung der Abspielverzögerung im Client das `PlayoutDelayBase` Interface vorgesehen. Es wird über `addPacketLatency()` mit der Latenz jedes vom Server empfangenen Antwortpaketes gespeist, die sich als Differenz der Sequenznummern des zuletzt aus der Audiohardware gelesenen und dem Antwortpaket ergibt. Die vorgestellten Variationen bei statistischen Berechnungen, Präferenzmatrizen und Bewertungsfunktionen werden in Form von Spezialisierungen dieser Basisklasse abgeleitet.

7.2.2.2 Echtzeit-Server RS

Analog zum RC lässt sich die Funktionsweise des Echtzeit-Servers RS grob vereinfacht als die eines Audio-Mischers im Netz beschreiben. Er wartet auf eingehende Audiopakete von allen teilnehmenden Clients,

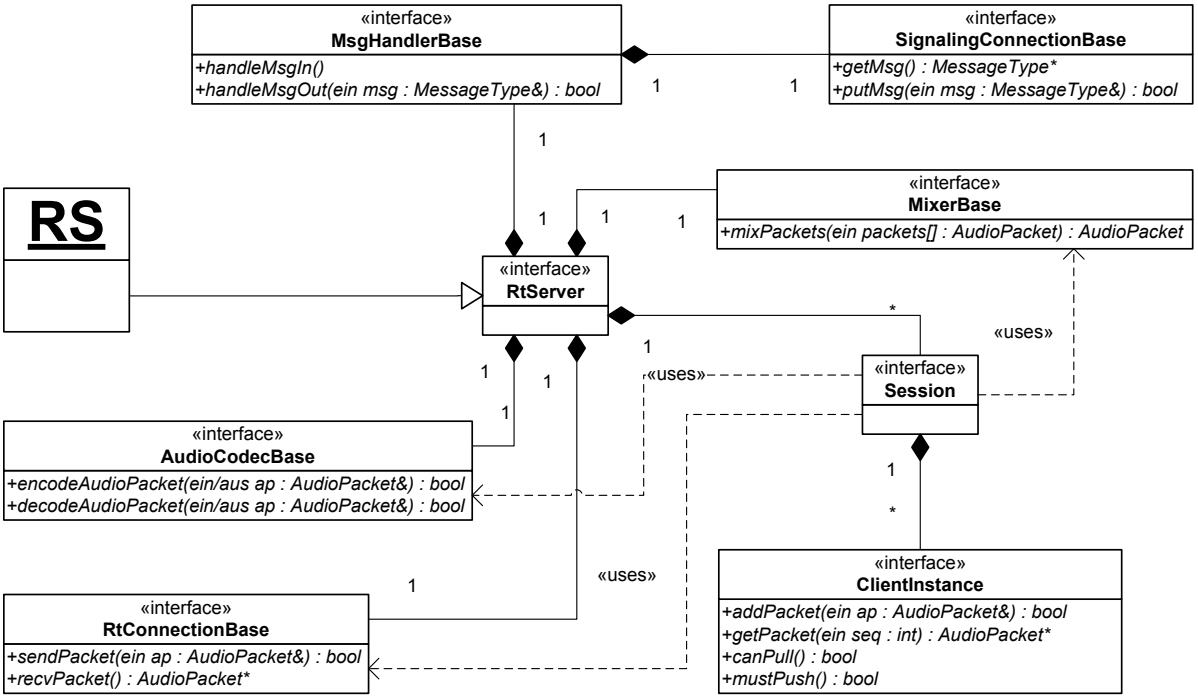

Abbildung 7.8: Klassendiagramm des Echtzeit-Servers RS

mischt zeitsynchrone Audiodaten und versendet eine Kopie der gemischten Audiopakete an jeden Teilnehmer.

Aus obiger Betrachtung zum Echtzeitverhalten beim RC ist auch der Betrieb des RS als Daemon naheliegend. Eine Realisierung als Callback innerhalb der ISR des Netzkarte ist dabei nicht möglich, da gemeinhin die Transportprotokolle im Kernel implementiert sind und auf Anwenderebene der Zugriff auf die über das Netz ausgetauschten Daten nur über die `socket` Schnittstelle möglich ist. Damit trotz der fehlenden Möglichkeit einer Anbindung im Interrupt Kontext mit minimaler Latenz und Jitter ausgeführt zu werden, muss der RS mit höchstmöglicher Priorität ausgeführt werden.

Die tatsächliche Herausforderung der hier vereinfacht dargestellten Trivialfunktion des Mischens von Audiodaten wurde in Abschnitt 6.2 detailliert untersucht und wird mit den dort ermittelten Ergebnissen im RS realisiert. Auch die Schlussfolgerungen der, ebenfalls in Kapitel 3 untersuchten, Herausforderungen hinsichtlich Fehlerverdeckung und Audiodatensynchronisation kommen im RS zur Anwendung.

Um die Verarbeitung mehrerer simultaner Sessions unterstützen zu können, muss der RS über eine rudimentäre Verwaltung von Nutzern, Gruppen und Sitzungen verfügen, um die verarbeiteten Audiodaten den zugehörigen Sitzungen zuordnen zu können. Über diese Minimalverwaltung hinaus gehende Managementfunktionalität und jegliche über das Echtzeit-Musizieren hinausgehende Weiterverarbeitung der Audiodaten wird im NC ausgelagert. Dieser kontrolliert über das Signalisierungsprotokoll NSRS den RS und greift über dieses Audiodaten und Statusinformationen für spätere On-Demand Verwendung ab.

Das Klassendiagramm für den RS ist in Abbildung 7.8 dargestellt. Die Kommunikationskanäle zu den benachbarten Komponenten sind analog zum RC jeweils über eine Instanz von `RtConnectionBase` und `SignalingConnectionBase` realisiert, wobei die Bearbeitung der Nachrichten des Signalisierungsprotokolls NSRS von einer `MsgHandlerBase` Instanz durchgeführt wird.

Der `RtServer` unterstützt die parallele Verarbeitung mehrerer Sitzungen, die er als Liste von `Session` Objekten aggregiert. Über den RCRS Kanal empfangene Audiopakete werden dabei anhand der Benutzer-

und Sitzungsidentifikation den aktiven Sessions zugeordnet. Eine Sitzung wiederum besteht aus ein oder mehreren Teilnehmern, die als Liste von `ClientInstance` Objekten realisiert ist. Ein vom `RtServer` erhaltenes Audiopaket leitet die `Session` an die jeweilige `ClientInstance` über die `addPacket()` Methode weiter, womit diese die Verwaltung aller Audiodaten übernimmt.

Innerhalb dieser Klasse sind ebenso die in Abschnitt 6.7 vorgestellten Mechanismen zur Bestimmung der Client spezifische De-Jitter Pufferlänge sowie die Umsetzung evtl. nötiger Maßnahmen zur Fehlerverdeckung enthalten. Die beiden Methoden `canPull()` und `mustPush()` entsprechen dabei den beschriebenen Richtlinien, anhand derer das `Session` Objekt die Mischzeitpunkte ermittelt. In einem solchen Fall sammelt `Session` von allen Clients über `getPacket()` das zu mischende Paket und versorgt ein `Mixer` Objekt mit dieser Liste. Das gemischte Paket wird anschließend über den RCRS Kanal an alle teilnehmenden Clients versendet.

Die gewählte Arbeitsweise des RS als passives System, welches ausschließlich über empfangene Audiopakete oder Nachrichten zur Signalisierung aktiv wird, vereinfacht und begünstigt das Design derart, dass zentrale Komponenten gemeinsam verwendet werden können: da der RS vollständig sequenziell abläuft und ohne Nebenläufigkeiten auskommt, ist je eine gemeinsame Instanz der Klassen `AudioCodecBase`, `MixerBase` und `RtConnectionBase` erforderlich, die von allen aktiven Sessions verwendet werden.

7.2.2.3 Zeitkritisches Transportprotokoll RCRS

Für die Übertragung der Audiodaten zwischen RC und RS haben wir das auf UDP basierende und an RTP angelehnte verbindungslose Protokoll RCRS entworfen. Der Bedarf für ein proprietäres Protokoll begründet sich aus den besonderen Betriebsparametern bei NMP einerseits und aus dem unpassenden RTP-Header andererseits. Zwar ist das RTP explizit für den Transport von Echtzeit-Multimediadaten bestimmt und unterstützt auch Kompositionen aus mehreren Einzelströmen. Die dabei eingesetzte Verwendung von Zeitstempeln zur Synchronisation deckt sich jedoch nicht mit unseren in Abschnitt 6.7 vorgestellten Verfahren für die Audiodatensynchronisation.

Darüber hinaus werden für den Abgleich zwischen RC und RS bei NMP zusätzliche Informationen ausgetauscht, die im RTP-Header nicht vorgesehen sind. Beispiele dafür sind Angaben über Versatzdrift beim Mischen, Positionsinformation für virtuelle Bühnen oder Hinweise zu im Audiopaket durchgeführten Fehlerverdeckungsmaßnahmen. Zwar lassen sich solche Informationen in der optionalen Header Extension des RTP-Headers oder als privater Header im Payload unterbringen, da aber die Pakete dann ausschließlich im NMP-System verwendbar sind, ist die Verwendung von RTP sinnlos.

Darüber hinaus vergrößert der RTP-Header jedes Datenpaket um mindestens 16 Bytes, was bei einer Paketrate von 375 Hz zu einer signifikanten Erhöhung des Datenaufkommens führt. Bei RCRS wurde aus dieser Tatsache heraus besonderen Wert darauf gelegt, den Nutzdaten vorangestellten Paketheader zu minimieren. Bei der Realisierung wurden dabei folgende Prinzipien angewandt:

Modularisierung

Alle Einträge sind optional und können bei Bedarf dem Paket hinzugefügt werden. Ein minimaler Kopf beinhaltet die Sequenznummer des Paketes und eine eindeutige Teilnehmer-ID. Optional können Informationen angehängt werden für

- Audiodatenformat
- Position auf der virtuellen Bühne
- Zeitmessungen
- Statistiken
- Kompositionsinformationen eines gemischten Audiopaketes

- Angaben über durchgeführte Maßnahmen zur Fehlerverdeckung

Die Komposition des Headers erfolgt als Liste von Sub-Headern, die durch ein in jeder Header-Komponente vorhandenes Ende Flag abgeschlossen wird.

Delta Kompression

Viele Header-Informationen sind über lange Perioden oder sogar über eine ganze Sitzung konstant. Zu den durchweg statischen Größen gehören dabei systembedingt das verwendete Audioformat und die Audioblockgrößen, die Netzadressen der Clients und seine Zugehörigkeit zur Sitzung. Größtenteils statisch sind Angaben über die virtuelle Bühnenposition eines Clients, das verwendete Audiokompressionsverfahren oder langfristige statistische Daten. Um die Übertragung solch statischer Daten zu minimieren, wird eine Delta Kompression verwendet, die dadurch ermöglicht wird, dass jedes versendete Paket nach sehr kurzer Zeit beantwortet wird. Sie beruht darauf, dass der Empfänger die mit einem Paket erhaltene Information dem Sender bestätigt und diesem zuverlässig den Kenntnisstand des Empfängers übermittelt. Eine statische Information wird nur dann im Paket-Header versendet, wenn diese von dem beim Empfänger vorhandene abweicht.

Der Mechanismus lässt sich am einfachsten am Beispiel des Audiodatenformates nachvollziehen: nach dem Beitritt zu einer Session sendet ein Client seine Audiodaten zusammen mit der Beschreibung des Datenformats zum Server. Sobald dieser den ersten Mischvorgang einleitet und das gemischte Paket zurücksendet, bestätigt er im Header den Empfang der Formatinformation. In der Zwischenzeit versendet der Client diese Information weiterhin mit jedem Audiopaket. Mit dem Empfang der ersten Bestätigung ist eine gesicherte Synchronisation dieser Parameter beim Client und Server nachgewiesen, eine weitere Übermittlung ist fortan nicht mehr nötig, solange das Datenformat gleich bleibt.

Auf Plausibilität und Lokalität gestützte Codierung

Verschiedene im Paket-Header übermittelten Werte und Betriebsparameter erfordern eine globale Eindeutigkeit und belegen damit einen großen Wertebereich. Bei der hohen Paketrate muss die Sequenznummer mit 32 Bit aufgelöst werden, gleiches gilt für System weit eindeutige Kennungen für Benutzer, Sitzung oder Gruppenzugehörigkeit: auch für diese Schlüssel sind Wertebereiche von 32 Bit vorgesehen. Da jedes vom Client ausgehende Audiopaket mindestens die Sequenznummer und die Client-ID mitführen muss, um beim Server richtig verarbeitet werden zu können, verursacht das Anfügen der unveränderten Werte einen Overhead von acht Bytes pro Paket. Stattdessen wird unter Berücksichtigung der Lokalität einer laufenden Sitzung der Wertebereich vor dem Senden reduziert und über eine Plausibilitätsprüfungen nach dem Empfangen rekonstruiert. Zwei Maßnahmen werden dabei im Wesentlichen zur Minimierung des Headers angewandt:

1. Aus der global eindeutigen ID für Nutzer, Gruppe und Sitzung berechnet der RC bei Eintritt eines Clients zur Session eine für ihn eindeutige Kennung, um den Datenstrom dieses Teilnehmers eindeutig identifizieren zu können. Angesichts technisch vorgegebener Einschränkungen von gleichzeitig aktiven Sitzungen und teilnehmenden Clients ist für eine zweifelsfreie Zuordnung ein Wertebereich von einem Byte völlig ausreichend.
2. Mit der Kenntnis, dass die Anzahl von Paketen auf dem Datenpfad zwischen der Erzeugung und dem Verbrauch der Audiodaten im Client auf maximal 30 beschränkt ist (was einer Systemlatenz von $30 \cdot t_\varphi = 80\,ms$ entspricht) wird deutlich, dass auch hier die Codierung der Sequenznummer mit einem Byte ausreichend ist, um diese rekonstruieren und eindeutig zuordnen zu können. Es genügt, wenn der Sender das niederwertigste Byte der Sequenznummer im Header mitführt, damit der Empfänger diese aufgrund ihres streng inkrementellen Charakters und detektierter Überläufe zuverlässig rekonstruieren kann.

Mit diesen Mechanismen erreichen wir für den überwiegenden Teil aller transportierten Datenpakete einen minimalen Header von zwei Bytes, der aus lokaler ID und Teilsequenznummer besteht.

Diese Minimierung des Headers und damit des Datenaufkommens erfolgt auf der Anwendungsebene des TCP-Schichtenmodells und ist nicht zu verwechseln mit Ansätzen für generische Header-Kompression wie bspw. die *ROHC* (RObust Header Compression, [12]), die über Vollzugriff auf den Protokollstack verfügt und somit in der Lage ist, die über den Verlauf einer Sitzung statische Felder im IP- und UDP-Header effizient zu komprimieren. Da, wie in Abschnitt 6.4 gezeigt, aufgrund der hohen Paketrate bei Verwendung von Audiodatenkompression die Protokoll-Header den Anteil der Nutzdaten je Paket signifikant verringern, kann die Verwendung von Header-Kompression eine deutliche Verbesserung dieses Missverhältnisses bewirken.

Die Anwendung solcher Verfahren erfordert eine durchgehende Unterstützung in den Netzprotokoll Implementierungen aller teilnehmenden Endsysteme, die es zum Entstehungszeitpunkt unseres NMP-Systems nicht gab. Der spätere Einsatz solcher Mechanismen bleibt durch die hier beschriebenen Optimierungen in jedem Fall unberührt.

Generell bleibt auch beim Entwurf der verwendeten Protokolle die Flexibilität durch einen modularen Ansatz gewahrt. Der gewählte Ansatz zugunsten bestehender Echtzeitprotokolle wie bspw. RTP ist der Pragmatik geschuldet: einerseits unterstützen diese die Verwendung der in NMP eingesetzten Audiocodecs nicht nativ, andererseits bedingt der generische Ansatz einen höheren Header Overhead und damit eine höhere Datenrate. In Szenarien, bei denen diese geringere Effizienz zweitrangig ist, lässt sich die Anbindung von RC und RS über RTP einfach realisieren, indem es über die definierte Schnittstelle angebunden wird.

7.2.3 Zeitunkritische Komponenten

Gemäß unserer Fokussierung auf die aus technischer Sicht herausfordernde Echtzeit-Funktionalität sollen in diesem Abschnitt die zeitunkritischen Komponenten vorgestellt werden. Wir beschränken uns dabei auf die Aspekte, die für das Verständnis der im folgenden Kapitel beschriebenen Implementationsvarianten erforderlich sind, eine detaillierte Abhandlung liegt dagegen außerhalb des Fokus dieser Arbeit.

7.2.3.1 Zeitunkritischer Server NS

Der NS ist als Komponente konzipiert, die multiple Instanzen von Echtzeit-Servern RS verwaltet und die anfallenden Audiodaten sammelt, um daraus höherwertige Dienste bereitzustellen. Zu den Kernaufgaben des NS gehören:

Zugangskontrolle zum System

Jeder Teilnehmer eines NMP-Systems muss global eindeutig identifizierbar sein, um diesen zu Sitzungen und archivierten Daten sicher zuordnen zu können. Dafür müssen Verwaltungsfunktionen für Nutzer, Sitzungen, Audiodaten und sonstigen Informationen vorhanden sein. Über diese kann sich der Musiker im System authentifizieren und neue Sitzungen definieren bzw. zu bereits erstellten beitreten. Unmittelbar nach dem Beitritt zu einer Sitzung wechselt der Teilnehmer in dem vom RS bereitgestellten Echtzeit-Modus. Während dieser Phase stellt der NS Funktionen zur Verarbeitung und Manipulationen an den Audiodaten bereit, bspw. um die Aufzeichnung zu initiieren.

Verwaltung der RS

Einem NS sind ein oder mehrere RS zugeordnet, die dieser über das NSRS-Protokoll kontrolliert und an die er durchzuführende Sitzungen delegiert. Für die günstige Auswahl eines RS benötigt der NS globale Metainformationen, die einerseits die Auslastung der einzelnen Echtzeit-Server als auch den für jede Sitzung optimalen RS umfassen. Diese Aspekte haben wir in NMP exemplarisch wie folgt umgesetzt: als Maß für die Auslastung eines RS wurde die Anzahl der simultanen Audiodatenströme verwendet, da

diese die Netzlast und die CPU-Auslastung, die i.W. durch Audiocodierung und -decodierung anfällt, widerspiegelt. Diese Kenngröße ist dem NS implizit bekannt, da jede Sitzung von diesem angestoßen und jeder Beitritt eines Musikers über diesen erfolgt.

Das zweite Entscheidungskriterium muss explizit ermittelt werden, da der für eine Sitzung optimale Server von Laufzeitgrößen abhängt und erst dann berechnet werden kann. In unserer exemplarischen Umsetzung sehen wir als optimalen Server denjenigen mit der minimalen Netzverzögerung zu allen Teilnehmern einer Sitzung an. Diese wird dabei ermittelt, indem der RC jedes Teilnehmers eine Pseudoverbindung mit allen in Frage kommenden Echtzeit-Servern aufbaut und die Netzverzögerung ermittelt. Aus diesen Messungen und den bekannten Auslastungen der RS wird ein geeigneter RS für die jeweilige Sitzung gewählt.

Sammeln und Archivieren von Musikdaten

Da dem RS selbst keine Zugriffe auf Speichermedien erlaubt sind, werden alle anfallenden Audiodaten über das Netz an den NS versendet, der diese indiziert und archiviert.

Bereitstellung von Audiospuren und Musikstücken

Mit den gesammelten Audiodaten lassen sich eine Vielzahl von Mehrwertdienste für den Benutzer realisieren, von denen wir exemplarisch zwei umgesetzt haben: Basis für eine direkte Unterstützung der Dienste ist eine Speicherung der Audiodaten in weitverbreiteten und von allen Betriebssystemen unterstützten Datenformaten. In unkomprimierter Form werden diese als WAV-Daten abgelegt, komprimiert liegen sie als MP3 bzw. AAC Dateien vor. Ein Herunterladen dieser Medien ist dann einfach über das FTP- oder HTT-Protokoll realisierbar, auch die Anbindung an Streaming-Server ist direkt möglich.

Unterstützung von On-Demand Szenarien

Neben den Abruf der Audiodaten zum späteren Anhören bietet unser System Unterstützung für so genannte On-Demand-Sitzungen, die dadurch definiert sind, dass mindestens ein Musiker durch seine vorher aufgezeichnete Audiospur vom System durch einen virtuellen Client ersetzt wird. Die Anwendungen dafür reichen vom temporären Ersatz eines verhinderten Band-Mitglieds bis hin zu Karaoke ähnlichen Szenarien, in denen ein Teilnehmer alleine sich einer virtuellen Sitzung anschließt. Die aufgezeichneten Daten werden dabei von einer oder mehreren Instanzen eines virtuellen Clients mit den Audiodaten der realen Teilnehmer gemischt.

Ein NS kann in kleineren NMP-Systemen ein einzelner Rechner sein, während in größeren Installationen eine Realisation als verteiltes System dedizierter Module erforderlich wird. Das für die Kommunikation zwischen NC und NS vorgesehene Protokoll NCNS ist so ausgelegt, dass jede Nachricht transparent zwischen einzelnen Komponenten des verteilten NS Systems ausgetauscht werden kann und sich keine Notwendigkeit für zusätzliche Protokolle ergeben. In Abbildung 7.9 ist ein solches System schematisch dargestellt.

Von dem für den Benutzer als Einzelrechner wahrnehmbaren Proxy werden die Anfragen an Verwaltungsrechner geleitet und über Load-Balancer an geeigneten Echtzeit-Servern Sitzungen initiiert. Die RS sind dabei geografisch verteilt, um von der meist vorhandene Lokalität von Mitgliedern einer Sitzung Nutzen zu ziehen und minimale Netzlatenzen zu gewährleisten. Die gesammelten Daten werden in ein oder mehrere Speichersysteme konzentriert, bevor ein zentraler Datenbankserver die im gesamten System anfallenden Audiodaten archiviert. Über FTP-, Web- und Streamingserver erfolgt für den Benutzer der Zugriff auf die gesammelten Daten.

Dieser Aufbau hat keine zeitkritischen Anforderungen und ermöglicht somit die beliebige Verteilung der Komponenten im Netz. Angesichts der anfallenden Datenmenge ist es jedoch Sinnvoll, die Knoten innerhalb eines Kern-Netzes zu betreiben und breitbandig miteinander zu verbinden. Nur so kann gewährleistet werden, dass die Verfügbarkeit der Daten zwischen den zeitkritischen und -unkritischen Phasen nahtlos erfolgt

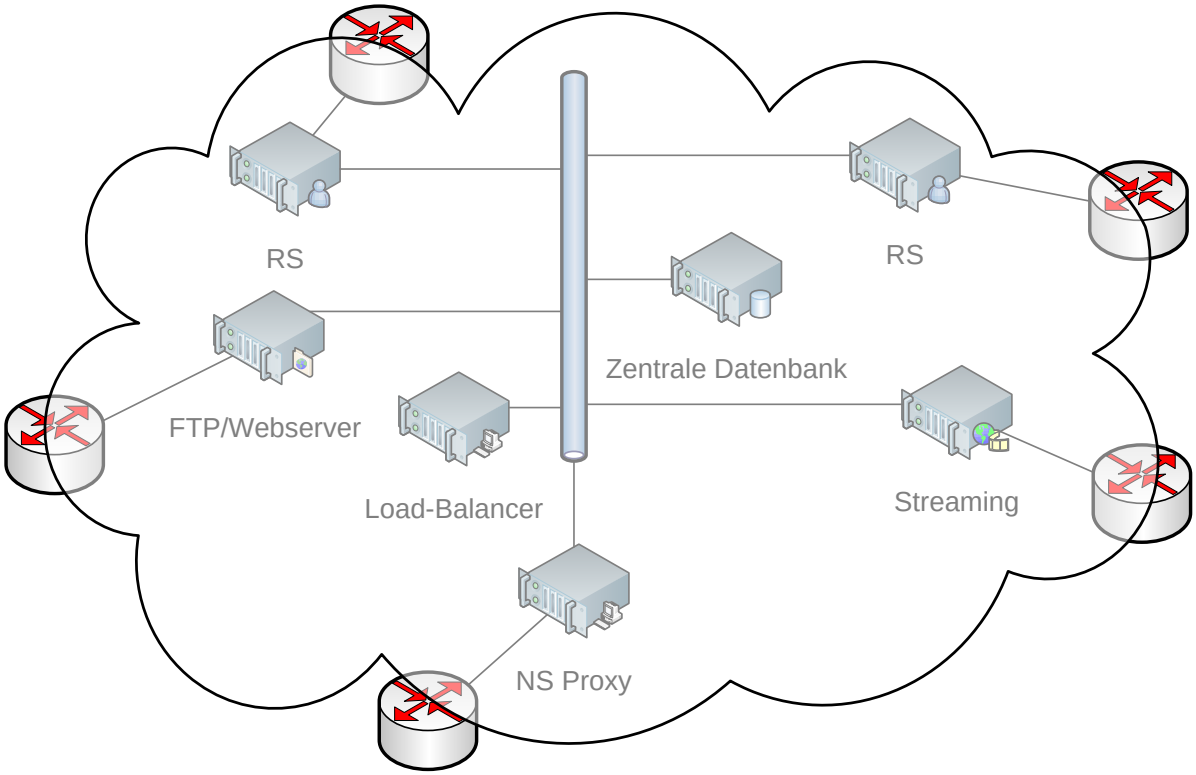

Abbildung 7.9: Realisierung des NS als verteiltes System

und die Musiker unmittelbar nach Abschluss einer Sitzung auf die aufgezeichneten Audiospuren zugreifen können.

7.2.3.2 Zeitunkritischer Client NC

Der NC ist als Bindeglied zwischen dem Benutzer und dem RC einerseits und als zentrale Verarbeitungseinheit von zeitunkritischen Daten andererseits konzipiert. Diese Komponente kann dann entfallen, wenn auf beide Funktionen verzichtet werden kann. So ist es beispielsweise möglich, die Schnittstelle zum Benutzer über ein Web-basiertes UI zu implementieren und die Konfiguration des RC serverseitig vom NS durchzuführen und das lokal Speichern der Audiodaten am Client zu unterbinden. Dieser Aufbau wurde umgesetzt und wird weiter unten in Abschnitt 8.2.3 genauer vorgestellt.

Ein direkter Zugriff vom Benutzer auf den RC kann wegen der beschriebenen Einschränkungen, die sich aus der Ausführung des Moduls innerhalb einer ISR ergeben, nur eingeschränkt erfolgen. Abläufe, die einen Kontextwechsel erfordern (dazu fallen grafische Benutzerführung, Zugriff auf blockierende Peripherie wie Festplatte, o.Ä.), dürfen nicht im RC realisiert werden und sind im NC ausgelagert. Der NC vereint somit alle Aufgaben, die nicht unmittelbar in den durch t_φ definierten zeitlichen Abständen vollständig abgearbeitet werden müssen. Dazu gehören

Zugang des Benutzers zum NMP-Server

Die Authentifizierung eines Musikers beim NMP-System und das Management seiner Sitzungen und Audiodaten ist der primäre Zweck des NC. Nach einer erfolgreichen Anmeldung kann der Musiker auf seine persönlichen Daten zugreifen und ändern, neue Sitzungen erstellen oder zu bestehenden beitreten. Die dafür vorgesehenen Nachrichten sind Teil des NCNS Protokolls.

Konfiguration des RC

Eine erfolgreiche Anfrage zum Beitritt zu einer Sitzung quittiert der NS mit den für die Kommunikation mit dem ausgewählten RS erforderlichen Zugangsdaten. Darunter fallen neben den Adressdaten auch die in Abschnitt 7.2.2.3 erläuterten Transformationen von globalen in lokale IDs zur Verringerung der zu übertragenden Paketheader. Mit den bereit gestellten Informationen beginnt der RC die Audiodatenübertragung zum RS und leitet damit die Sitzung in die Echtzeit-Phase über.

Benutzerschnittstelle

Die gesamte Client-seitige Interaktion des Benutzers erfolgt über den NC, da ein direkter Zugang zum RC nicht möglich ist. Die Realisierung der UI ist dabei keinen Einschränkungen unterworfen, da die Ausführung von Aktionen vom NC zum RC und die Rückmeldung von Statusinformationen in der Gegenrichtung vollständig über den Austausch von Nachrichten über das NCRC-Protokoll erfolgen. Wir haben für NMP rein textuelle, graphische und webbasierte Benutzerschnittstellen implementiert, die wir im nächsten Kapitel vorstellen werden.

Aufzeichnung der lokalen Audiodaten

Eine Client-seitige Aufzeichnung der lokalen Audiodaten ist für eine lückenlose Archivierung aller Audiospuren im Server erforderlich für die Fälle, in denen Audiopakete auf ihrem Weg vom RC zum RS verworfen werden (d.h., Paketverluste ohne verspätete Pakete). In dem Fall kann der NS bei der Detektion fehlender Daten die Pakete mit den betreffenden Sequenznummern vom NC erneut anfordern und die Lücken nachträglich schließen. Die Übertragung der Audiopaketkopien vom RC zum NC ist Teil des RCNC Protokolls.

Auch für den Client gilt, dass die beschriebenen Funktionen einschließlich der RC Komponente von einem einzelnen Rechner abgearbeitet werden, oder aber auf mehrere Knoten eines verteilten Systems delegiert werden können. Wir haben dabei Szenarien umgesetzt, bei denen der RC von einem eingebetteten System, die Benutzerführung über ein PDA und die Audiodatenaufzeichnung von einem PC bereitgestellt wird.

7.2.3.3 Signalisierungsprotokolle NCNS, NCRC und NSRS

Die dargestellte Flexibilität bei der Verteilung von Funktionseinheiten im Netz wird zum großen Teil durch den Aufbau der verwendeten Protokolle zur Signalisierung der Komponenten untereinander erreicht.

Die eingesetzten Signalisierungsprotokolle NCNS, NCRC und NSRS setzen auf TCP auf und sind daher verbindungsorientiert. Sie folgen einem streng sequenziellen Anfrage-Quittierungs Schema, in dem jede Komponente mit der Gegenseite über definierte Nachrichten eine Aktion anstößt und mit dem Empfang der Bestätigung abschließt.

Da die zeitunkritischen Komponenten und damit auch die Kommunikation zwischen diesen nicht Kern dieser Arbeit ist, haben wir bei der Entwicklung des Protokolls das KISS Prinzip (*keep it small and simple*) angewandt und den Fokus auf Einfachheit bei Umsetzung und Überprüfbarkeit gelegt. Das Resultat ist ein flexibles, leicht gewichtiges und sehr einfach anzuwendendes Klartextprotokoll, mit dem auf einfachste Weise Nachrichtenobjekte definiert, versendet und rekonstruiert werden können.

In unserem Entwurf bilden wir in Nachrichten zu übertragende Datenstrukturen in Form von Key-Value Paaren ab. Basis für die Formatierung dieser Nachrichten bilden Tag-Objekte, die in Anlehnung an Beschreibungssprachen wie HTML oder XML aus einem öffnenden und einem schließenden Tag im Datenstrom bestehen. Den Tag-Bezeichner interpretieren wir als Key, während der Value zwischen den Elementen zu finden ist. Ein Value kann dabei ein Wert sein, oder wiederum aus anderen Tag-Objekten bestehen. Mit diesem

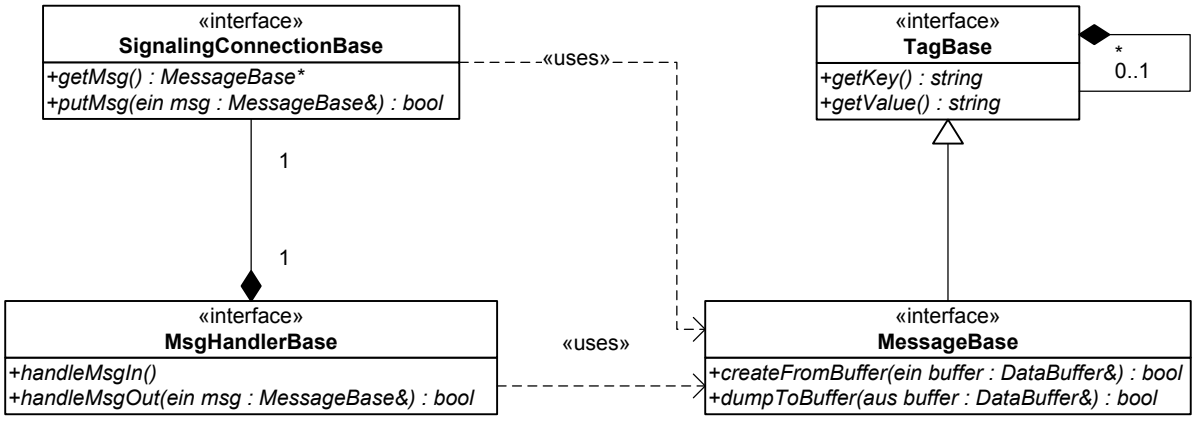

Abbildung 7.10: Aufbau der Nachrichten für Signalisierungsprotokolle

Ansatz lassen sich sehr einfach beliebige Datenstrukturen zwischen Komponenten austauschen, wie am folgenden Beispiel für die Übertragung der Login-Daten eines Benutzers verdeutlicht wird:

Struktur	Tag-Format

`LoginData = {`
`  name = "bob"`
`  password = "bob4nmp4"`
`}`
	`<LOGIN_DATA>`
	`  <NAME>bob</NAME>`
	`  <PASSWORD>bob4nmp4</PASSWORD>`
	`</LOGIN_DATA>`

Nachrichtenobjekte werden in diesem Schema wie in Abbildung 7.10 dargestellt so abgelegt, dass dem Namen der Nachricht als Key alle erforderlichen Parameter als Value in Form eines Tag-Objektes folgen. Jedes Nachrichtenobjekt ist in der Lage, mit der Methode `dumpToBuffer()` seine Daten im beschriebenen Tag-Format in ein `DataBuffer` zu serialisieren, welcher über einen Nachrichtenkanal übermittelt wird. Auf der Gegenseite werden die Nachrichtenobjekte über `createFromBuffer()` direkt aus dem empfangenen Datenstrom zurückgewonnen.

Die Schnittstelle zum Übertragungskanal stellt `SignalingConnectionBase` bereit, welche selbst auf der bereits beschriebenen `TcpSocket` Klasse aufsetzt. Sobald dieser Daten empfängt, versucht das angebundene `SignalingConnection` Objekt über `getMsg()` eine Nachricht zu rekonstruieren. Ist im Empfangspuffer mindestens eine Nachricht vollständig enthalten, wird ein `MessageBase` Objekt erzeugt und von einer `MsgHandlerBase` Instanz innerhalb `handleMsgIn()` abgearbeitet.

Ausgehende Nachrichten werden dem Message Handler über `handleMsgOut()` zugeführt, die er über die Nachrichtenmethode `dumpToBuffer()` in den Ausgangsdatenstrom schreibt, der abschließend von `SignalingConnectionBase` per `putMsg()` übertragen wird.

Mit diesem Mechanismus wird die Bearbeitungslogik für Nachrichten in die Nachrichtenobjekte selbst gekapselt, wodurch sie nicht mehr an ein übergeordnetes Endsystem gebunden sind und somit die Flexibilität bei der Verteilung von Funktionseinheiten auf unterschiedliche Netzknoten ausmachen. So kann der Message Handler in jedem Knoten wie ein Proxy agieren und empfangene Nachrichten entweder selbst verarbeiten, wenn er über die erforderlichen Funktionen verfügt, oder diese an seine Nachbarn weiterleiten.

Auf eine detaillierte Diskussion der Protokollspezifikation wird hier mit Verweis auf die Dokumentation im NMP Quellcode verzichtet. Stattdessen werden anhand des in Abbildung 7.11 skizzierten typischen Ab-

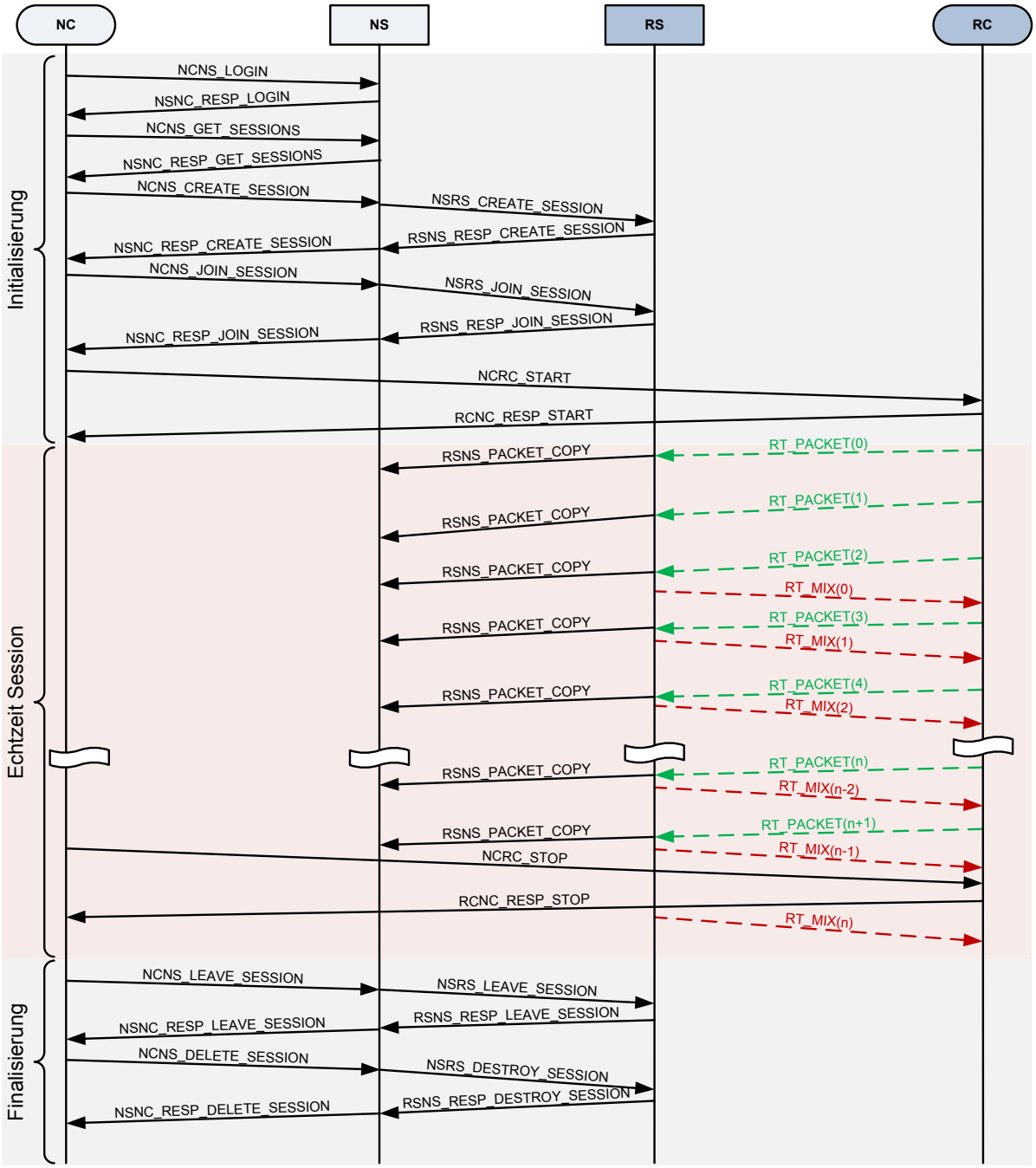

Abbildung 7.11: Typischer Protokollablauf während einer NMP-Sitzung

laufs während einer NMP-Sitzung die wichtigen Phasen und beteiligten Nachrichten exemplarisch beschrieben.

Der Ablauf einer NMP Sitzung lässt sich grob in drei Abschnitten unterteilen:

Initialisierung

In der ersten Phase, die ausschließlich zwischen NC und NS verläuft, erlangt der Benutzer Zugang zum System und bereitet die Sitzung vor. Die Authentifizierung wird mit einer `NCNS_LOGIN` Nachricht vom NC initiiert, die der NS mit seiner Antwort in Form einer `NSNC_RESP_LOGIN` beantwortet. Fällt die Zugangsprüfung erfolgreich aus, enthält die Quittierung die für diese Sitzung gültigen Kennungen, mit

denen der Benutzer alle folgenden Nachrichten zur Validierung kennzeichnen muss.

Der anschließende zweite Schritt ist der Beitritt zu einer Sitzung. Der Benutzer kann über NCNS_GET_SESSIONS eine Liste aller derzeit im Server bekannten Sitzungen abfragen und, falls keine für ihn passende vorhanden ist, über NCNS_CREATE_SESSION eine neue erstellen. Für die neu erstellte Sitzung wählt der NS aus den ihn zugeordneten RS den optimalen aus und erstellt dort über NSRS_CREATE_SESSION eine neue Sitzung. Die zum Zugang erforderlichen Kennungen beantwortet der RS mit RSNS_RESP_CREATE_SESSION. Der NS ergänzt diese um die Adresse des RS und macht diese dem Benutzer über NSNC_RESP_CREATE_SESSION bekannt.

Mit diesen Informationen kann der Musikern anschließend seinen Beitrittswunsch mit dem NCNS_JOIN_SESSION Kommando äußern. Der NS leitet diesen Wunsch an den RS weiter, der mit einer positiven Nachricht bestätigt, ab sofort vom betreffenden Client empfangene Audiopakete zu verarbeiten und der gewählten Sitzung zuzuordnen. Sobald der NC diese vom NS weitergeleitete Bestätigung in Form einer NSNC_RESP_JOIN_SESSION Antwort erhält, weist er seinen RC über das Kommando NCRC_START an, die kontinuierliche Audiodatenkommunikation mit dem RS zu initiieren und damit in die Echtzeitphase der Sitzung zu wechseln.

Echtzeit-Sitzung

In dieser Phase des Netzverteilten Musizierens erfolgt der kontinuierliche und autarke Audiodatenaustausch zwischen den Echtzeitkomponenten RC und RS. Die Benutzerinteraktion mit dem System ist nahezu vollständig auf das Musizieren beschränkt, da während der Echtzeit-Sitzung alle Betriebsparameter konstant sind. Lediglich das Starten und Stoppen der Server-seitigen Aufnahmefunktion ist als vom Teilnehmer auszulösende Funktion vorgesehen. Die für eine erfolgreiche Durchführung sonstigen erforderlichen Kommunikationen zwischen den Komponenten (wie Austausch von Statusinformationen, Abgleich von Audiodaten, Routing von Nachrichten, etc.) erfolgen ohne Eingriff des Musikers.

Finalisierung

Der Abschluss einer Echtzeit-Sitzung wird vom Benutzer über den NC initiiert und durch eine Signalisierung am Server und am RC durchgeführt. Im ersten Schritt fordert der NC dabei den RC über NCRC_STOP auf, den Datenaustausch mit dem RS zu beenden. Anschließend teilt er dem NS über NCNS_LEAVE_SESSION mit, dass er die Sitzung verlässt. Diese Information erreicht als Weiterleitung auch der RS und beendet die Assoziierung des Musikers mit der laufenden Sitzung. Damit ist die Echtzeit-Sitzung für diesen Teilnehmer formal abgeschlossen. Der Benutzer kann in dieser abschließenden Phase, solange er beim Server noch angemeldet ist, unmittelbar auf die aufgezeichneten Daten und Sitzungen zugreifen.

Analog zum Entwurf des Echtzeitprotokolls RCRS gilt auch hier, dass er auf die besonderen Anforderungen für die Signalisierung in NMP zwar eingeht, gleichwohl eine generelle Austauschbarkeit unterstützt. Der strikt modulare Charakter bleibt erhalten und unterstützt den Ersatz unserer proprietären Signalisierung durch etablierte Verfahren. Zur Integration in bereits bestehende Systeme sind so Anbindungen über bspw. SOAP [63], RPC [88] oder SIP [79] denkbar, die über Spezialisierungen der vorgestellten Schnittstellen zu realisieren sind.

7.3 Zusammenfassung

Eine Realisierung der analytisch ermittelten Latenzschranke erfordert eine Trennung zwischen zeitkritischen von -unkritischen Komponenten beim NMP-Client und Server. In unserem Grobkonzept erfüllen der RC und der RS alle Funktionen für das interaktive Musizieren, die über das verbindungslose RCRS-Protokoll Audio-

daten austauschen. Alle sonstigen Funktionen, die über diesen Basisdienst hinausgehende Mehrwertdienste anbieten, sind in den Komponenten NC und NS ausgelagert.

Die Interaktion zwischen den Komponenten wird von verbindungsorientierten Signalisierungsprotokollen bereitgestellt. Über das NCRC-Protokoll kontrolliert der NC den RC, indem er Benutzereingaben zu ihm weiterleitet und seine Konfiguration übernimmt. Analog dazu wird das NSRS-Protokoll zur Steuerung von auf RS ablaufenden Sitzungen durch den NS verwendet. Den Zugriff des Benutzers zum NMP-Server stellt das NCNS-Protokoll bereit, über dem die Signalisierung zwischen NC und NS läuft.

Der Entwurf der Signalisierungsprotokolle basiert auf einfache Tag-formatierte Nachrichtenobjekte, in denen die Auswertung und Bearbeitung der jeweiligen Nachricht gekapselt ist. So können Funktionseinheiten transparent modularisiert und auf beliebige Systeme im Netz verteilt werden.

Die vorgestellte SW-Architektur folgt dem Komponentendesign und wurde im Hinblick auf folgende Anforderung entworfen:

Plattformunabhängigkeit

Für die Einhaltung der Latenzschranke müssen die zeitkritischen Komponenten auf System naher Ebene auf die Hardware zugreifen, womit keine vollständige Plattformunabhängigkeit erreichbar ist. Der System spezifische Teil wird in den Schnittstellen `AudioIoBase` und `SocketBase` zum Zugriff auf Audio- bzw. Netzhardware gekapselt. Alle sonstigen Betriebssystem spezifischen Funktionen werden über die abstrakte `OSALBase` Schnittstelle bereitgestellt. Unter Verwendung dieser drei Interfaces sind alle übrigen SW-Module unabhängig vom darunter liegendem System bzw. Hardware. In NMP sind die genannten Schnittstellen für die Arbeitsplatz Systeme Microsoft Windows, Apple OS/X und Linux implementiert. Daneben wurden exemplarisch Anbindungen an eingebettete Systeme mit Echtzeitbetriebssystem mit jeweils angepassten Spezialisierungen der genannten Schnittstellen realisiert.

Erweiterbarkeit

Analog zu den Schnittstellen zur Systemabstraktion gibt es für jedes Objekt innerhalb unseres NMP-Systems ein abstraktes Interface, wodurch eine Erweiterbarkeit der verwendeten Klassen gewährleistet wird. Für die Verwendung eines künftigen Audiocodecs in NMP ist so bspw. nur die Implementierung einer Schnittstelle `AudioCodecBase` erforderlich, die die jeweilige Datenverarbeitung des Codecs in das vorgesehene blockbasierte Schema wandelt.

Flexibilität

Die einfache Erweiterbarkeit und Austauschbarkeit führt implizit zu einer hohen Flexibilität des Systems. Eine Auswahl von Funktionsblöcken und deren Verteilung auf unterschiedliche Module ist transparent durch die Instantiierung der verfügbaren Klassen direkt aus der Quellcodebasis möglich. Ein auf dem Arbeitsplatzrechner mit vollem Funktionsumfang laufender RC und ein auf eingebetteter Hardware laufende Minimalvariante sind Spezialisierungen der gleichen Basisklasse und werden innerhalb des NMP-Systems gleichbehandelt. Die Realisierung beliebiger Varianten aller Komponenten ist ohne Modifikation des Quellcodes über Instantiierung und Assoziation der beteiligten Objekte möglich.

Skalierbarkeit

Zusammen mit der Flexibilität bei der Modularisierung ermöglicht die native Unterstützung von Weiterleitungsmechanismen bei den entworfenen Signalisierungsprotokollen eine größtmögliche Skalierbarkeit beim Aufbau von NMP-Systemen. Eine definierte Komponente muss dabei entweder in der Lage sein, eine an sie gerichtete Nachricht verarbeiten zu können, oder diese an die richtige Stelle weiterzuleiten. So lassen sich beliebig komplexe Systeme aufbauen – die Spannweite reicht von einem vollständig auf einen einzelnen Rechner laufenden System bis hin zu einem globalen System mit weltweit verteilten Zugangsknoten.

Einen Einblick darüber, welche Möglichkeiten diese Eigenschaften beim Aufbau von NMP-Systemen erlauben, geben wir im nächsten Kapitel. Wir stellen verschiedene Modularisierungsgrade anhand ausgewählter Implementierungen von NMP Varianten vor, die wir im Projektverlauf um- und eingesetzt haben.

KAPITEL 8

IMPLEMENTIERUNG

Das vorgestellte Design unserer NMP-Software ist das vorläufige Ergebnis iterativer Anpassungen und Verbesserungen am initialen Prototypen zu Beginn unserer Forschungstätigkeit. Die Modifikationen wurden wechselseitig von Entwurf und Implementierungsphasen getrieben, indem unterschiedliche NMP Varianten auf Basis des zum Entstehungszeitpunktes gültigen Designs realisiert und generalisierbare Komponenten ins Design zurückgeflossen sind.

Das vorgestellte SW-Konzept als Resultat von Generalisierungen bestehender Implementierungen ist somit implizit mit den aufgeführten Eigenschaften behaftet und als Basis für verschiedene Realisationsvarianten verwendbar.

In diesem Abschnitt stellen wir einige dieser Varianten vor, die in verschiedenen Projektabschnitten und für unterschiedliche Zwecke implementiert wurden. Einleitend werden die externen SW-Bibliotheken und Komponenten aufgelistet, die wir in NMP verwenden.

8.1 Externe Bibliotheken

Bei der Implementierung von NMP haben wir nach Möglichkeit auf existierende Bibliotheken aufgesetzt. Zum Einsatz kommen dabei folgende externe SW-Komponenten:

ASIO SDK

Für den latenzarmen Zugriff von Windows und Apple-Rechnern auf Audiohardware hat der Hersteller Steinberg mit ASIO (*Audio Stream Input/Output*) einen Quasi-Standard etabliert, der praktisch von allen Soundkarten aus dem semi- und professionellen Bereich unterstützt wird. Für eine zuverlässige Einhaltung der Systemlatenz erfolgt auf Windows NMP-Clients der Zugriff auf die Audiohardware über diese Schnittstelle, die auf Anwendungsebene über das ASIO-SDK [89] angesprochen wird. Dieses stellt das Unternehmen nach einer vormaligen Registrierung über seine Webseite kostenlos zur Verfügung. In NMP wird das SDK 2.0 verwendet.

RtAudio

Die SW-Bibliothek *RtAudio* [82] stellt einen Software Wrapper für den abstrakten Zugriff auf plattformspezifische Audioschnittstellen von Windows, Mac OS/X und Linux bereit. Sie wurde von Garry P. Scavone als im Rahmen seiner Forschungsarbeit an der McGill Universität Montreal entwickelt und als

OpenSource freigegeben. Die SW-Bibliothek ist explizit für latenzkritische Audioanwendungen ausgelegt und deckt idealerweise in NMP Zugriffe auf alle Audioschnittstellen ab. Tatsächlich wird ein zuverlässiger Betrieb bei den für NMP erforderlichen Paketgrößen unter Linux und Mac OS/X erreicht, unter Windows treten dagegen sporadische Störungen auf. Wir verwenden daher RtAudio für den Zugriff über ALSA und CoreAudio auf Linux bzw. Mac OS/X Rechner und greifen auf Windows direkt auf ASIO zu. Verwendung findet in NMP die Version 3.0.3 vom 18. November 2005.

libFLAC

FLAC (Free Lossless Audio Codec) [97] gehört heute zu den populärsten verlustlosen Audiokompressionsverfahren, der Dank seiner freien Verfügbarkeit, Patentfreiheit und hoher Effizienz zum Quasi-Standard für die Archivierung von Audio-CDs und den Austausch von Audiodaten allgemein geworden ist. Neben dem Wegbereiter MP3 spielen heute viele Multimediageräte FLAC-Dateien nativ ab, gleichzeitig bieten Labels und Musiker den Download von Alben FLAC-codiert an. In NMP setzen wir die FLAC-Bibliothek libFLAC unmodifiziert in der Version 1.1.2 aus dem Jahre 2005 ein.

libWavPack

Der standardmäßig in NMP verwendete Audiocodec ist der von David Bryant entwickelte und unter OpenSource gestellte *WavPack* [23]. Ebenfalls als verlustloses Verfahren konzipiert bietet es gegenüber FLAC den hybriden Modus, in dem der codierte Datenstrom in einem Basislayer mit definierbarer Kompressionsrate und einem dazu passenden Korrekturlayer aufgeteilt wird. Nur mit der Verwendung des Basislayers können wir in NMP sicherstellen, dass die vorhandene Netzkapazität nicht überschritten wird, während der Korrekturlayer für die verlustlose Rekonstruktion der originalen Daten außerhalb der Echtzeitsitzungen übermittelt werden kann. Für den Einsatz in NMP haben wir die in Abschnitt 6.4.4.4 beschriebenen Modifikationen an der libWavPack Version 4.40 vom Dezember 2006 vorgenommen.

libAppWeb

Die Realisierung Server-seitiger Signalisierung und webbasierter Benutzerschnittstellen wurde mit Hilfe der quelloffenen *libAppWeb* [27] umgesetzt. Diese Bibliothek stellt einen eingebetteten Webserver bereit, der neben den üblichen Abruf von Webseiten nativ *Embedded Server Pages* (ESP) unterstützt. Diese können im HTML-Text durch entsprechende Tags eingebettete Funktionen erhalten, die vom Hauptprogramm direkt abgearbeitet werden und somit hoch reaktive und dynamische Benutzerschnittstellen ermöglichen. Für NMP ist dieser Mechanismus für den Funktionsnachweis eines Server-seitigen Betriebs des Systems ideal, da sich die über den Browser getätigten Eingaben direkt den entsprechenden Signalisierungen zuordnen lassen und zur Ausführung gebracht werden können. Zur Verwendung kommt bei NMP die libAppWeb in der Version 2.0.4 vom September 2005.

LIVE555 Streaming Media Library

Für den On-Demand-Modus, also den Abruf aufgezeichneter Audiospuren, werden in verschiedenen NMP Realisierungen die Dienste Download und Streaming angeboten. In den Fällen, in denen das Streaming der Audiodaten zum Client als integraler Teil des Systems bereitgestellt wird, greifen wir auf die *LIVE555* Bibliothek [56] zu. Diese im Quellcode frei verfügbare Komponente realisiert das Streaming von Mediadaten über RTP/RTSP und wird in vielen OpenSource Projekten verwendet (bspw. VLC, mplayer). In NMP setzen wir die Version vom 27.10.2006 ein.

Darwin Streaming Server

In modularen und verteilten NMP-Szenarien mit dedizierten Streaming Server setzen wir dagegen auf den *DarwinStreamingServer* (DSS) [5], der OpenSource Variante von Apples QuickTime Server. Da eine Instanz vom DSS innerhalb eines NMP-Systems nur über die aufgezeichneten Mediadaten interagiert,

gibt es keine Abhängigkeiten hinsichtlich der verwendeten Version. Wir haben zuletzt die DSS Version 5.5.5 vom Dezember 2007 verwendet.

wxWidgets

Für die Realisierung graphischer Benutzerschnittstellen haben wir auf das plattformübergreifende Framework *wxWidgets* [87] zurückgegriffen. Damit ist die Quellcode kompatible Entwicklung von GUI Anwendungen für verschiedene Zielsysteme möglich, u.a. für Windows, Linux und Mac OS/X. Wir haben zum Entstehungszeitpunkt diesem Framework den Vorzug gegenüber dem weiter verbreiteten *Qt* [9] von Trolltec (heute zu Nokia gehörend) gegeben, da dieses erst später für die Verwendung mit Windows unter einer OpenSource Lizenz gestellt wurde. In den implementierten NMP Varianten wurde die zum damaligen Zeitpunkt aktuelle wxWidgets Version 2.6.3 verwendet.

8.2 Realisierte NMP-Varianten

Im Projektverlauf wurden am IBR einige unterschiedliche Varianten von NMP implementiert. Von diesen heben wir in diesem Abschnitt drei hervor, mit denen die Eigenschaften des SW-Designs im Hinblick auf Modularität und Flexibilität nachgewiesen wird.

8.2.1 Minimales Entwicklungs- und Testsystem

Die Aufbau eines NMP-Systems aus netzverteilten Komponenten stellt in der Implementierungs- und Testphase eine eher hinderliche Eigenschaft der Architektur dar, da das Aufsetzen eines vollständigen Systems zeitaufwendig ist und das Risiko variabler Testbedingungen mit sich bringt. Während der Entwicklung wurde zu diesem Zweck ein minimales NMP-System verwendet, welches vollständig auf einem einzelnen Rechner läuft und im Wesentlichen folgendem Ablauf genügt:

1. Aufsetzen eines RC
2. Aufsetzen eines RS
3. Starten einer Pseudo-Sitzung
   a) Erzeugen einer Pseudo-Sitzung beim RS
   b) Hinzufügen des RC zur Sitzung
   c) optional: hinzufügen virtueller Clients zur Sitzung
   d) Starten des RC

In einem derart aufgesetzten System liegt der Fokus auf die Echtzeit-Sitzung, während die zur Durchführung erforderlichen Auf- und Abbauphasen auf das Nötigste beschränkt sind. Die Statusausgaben eines solchen Minimalsystems sind in Abbildung 8.1 dargestellt. Zu sehen sind die im Client und Server berechneten Statistiken, aus denen die Ermittelung der Abspielverzögerung beim RC gemäß Abschnitt 6.5 berechnet wird, sowie der Zustand der Mischer Queues im RS, wie in Abschnitt 6.7 beschrieben.

Für das Aufsetzen eines solchen NMP Aufbaus sind je eine Instanz des RC und RS erforderlich, während für deren Signalisierung eine Programm implementiert wurde, welches die erforderlichen Nachrichten für Initiierung und Terminierung von Sitzungen vom NC und NS bearbeiten kann. Dieser Aufbau und der damit resultierende Protokollfluss sind im Diagramm 8.2 skizziert. Die Signalisierungskomponente ist ihrer Anwendung bei der Entwicklung gemäß als Debug NC/NS bezeichnet. Sie startet eine Echtzeit-Sitzung zwischen den bereits ausgeführten Komponenten RC und RS durch die aus Abschnitt 7.2.3.3 bekannten Nachrichten zum

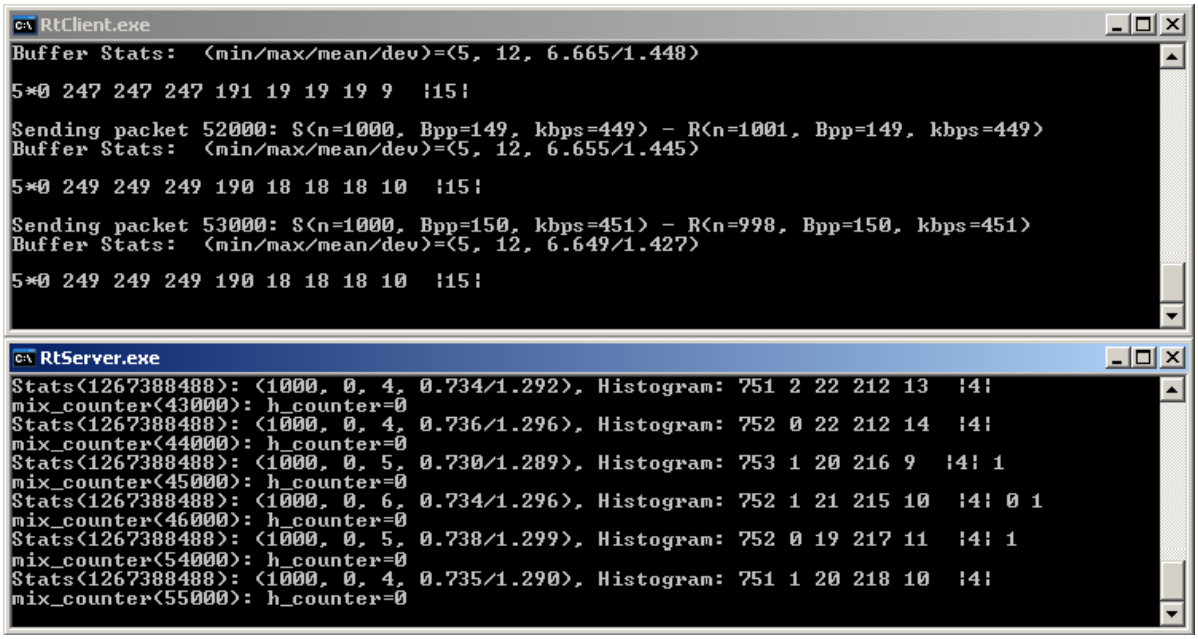

Abbildung 8.1: Programmausgabe der Komponenten während einer Debug-Sitzung

Erzeugen einer Sitzung beim RS und Hinzufügen des RC zur Teilnehmerliste dieser Sitzung. Mit einem anschließenden Start-Kommando an den RC geht die Sitzung in die Echtzeit-Phase über und wird zuletzt mit den dafür vorgesehenen Nachrichten wieder abgebaut.

Dieser Aufbau kann mit all seinen Komponenten vollständig in einem Rechner erfolgen, wenn es bspw. um die Prüfung der Interaktionswege zwischen diesen geht. Auch für Testaufbauten zur Evaluierung des NMP-Systems lässt sich diese Pseudo-Komponente verwenden. Da für Musiker im Wesentlichen die Leistung in Live Sessions relevant ist und das Aufsetzen der Tests unter der Leitung der Entwickler durchgeführt werden, eignet sich die Debug NC/NS Komponente zum Anstoßen von Sitzungen sehr gut.

Generell konnten wir diesen Aufbau für alle Szenarien wählen, in dem nur die Echtzeit-Funktionalität von NMP relevant ist – was für die interne Projektarbeit nahezu durchgängig zutreffend gewesen ist.

8.2.2 Client kontrollierte GUI-Anwendung

Für den Betrieb von NMP durch fachfremde Musiker und zur allgemeinen Demonstration der NMP-Software wurde ein Zugang mit graphischer Benutzerschnittstelle implementiert. Diese auf wxWidgets aufsetzende und für Windows, Mac OS/X und Linux verfügbare Programmkomponente ist teil des zeitunkritischen Clients NC. Sie ermöglicht die Konfiguration des RC und stellt den Zugang zum NMP Dienst bereit. Alle für die Verwaltung auf und Durchführung von Sitzungen erforderlichen Funktionen können über diesen Client ausgeführt werden, gleichzeitig bezieht er vom RC statistische und momentane Zustandsdaten, um dem Benutzer ein zeitnahes visuelles Feedback geben zu können.

In Abbildung 8.3 ist ein Screenshot der Client GUI während einer laufenden Sitzung dargestellt. Die im oberen Teil enthaltene Pegelanzeige für Audio-Ein- und Ausgabe wird anhand von Intensitätsmessungen des RC für die von der Audiohardware ausgelesenen und vom RS empfangenen und auszugebenden Audiodaten gebildet. Der RC ermittelt dabei die Werte für minimalen, maximalen und mittleren Ausschlag innerhalb Messperioden von 20 ms und überträgt diese Werte kontinuierlich zum NC. Die resultierende Feedback ist einerseits ein hilfreiches Instrument für den Musiker und andererseits ein zuverlässiger Indikator bei Fehl-

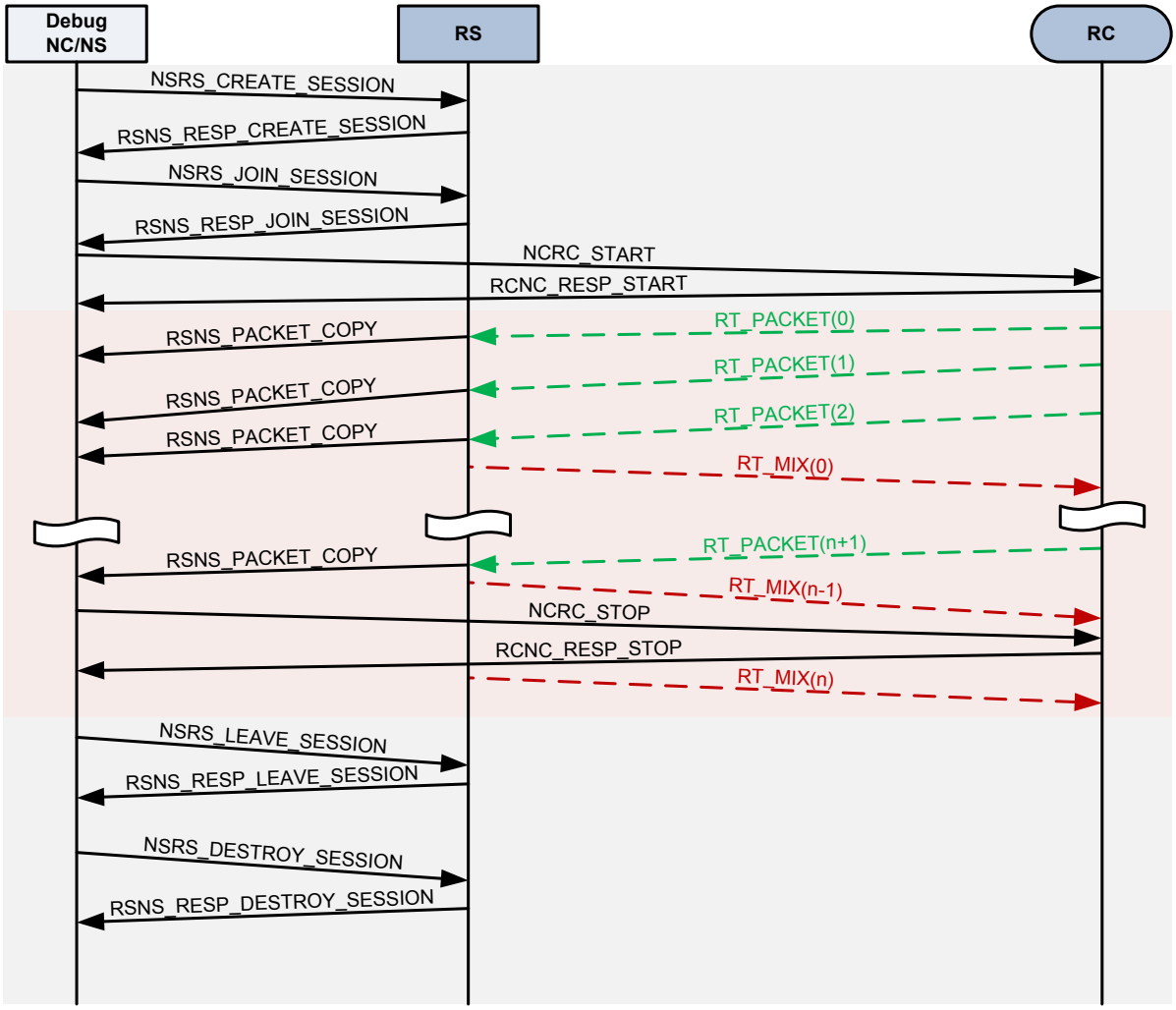

Abbildung 8.2: Aufsetzen einer NMP-Sitzung mit minimaler Signalisierung

funktionen des NMP-Systems: da die Status der Audiodaten an den Eingängen der Audio und Netzschnittstelle dargestellt werden, sind Informationen über den gesamten Datenpfad verfügbar, so dass potentielle Probleme im Audioteil und auf der Netzseite gleichermaßen erkannt werden können.

Unter der Pegelanzeige sind die Buttons für die während einer Echtzeit-Sitzung ausführbaren Funktionen platziert, die sich darauf beschränken, die laufende Sitzung zu verlassen oder die Aufzeichnung der Audiodaten am NS zu aktivieren. Die im NS bisher aufgezeichneten und für einen Teilnehmer abrufbaren Tonspuren sind in in der Liste enthalten, die den Hauptteil grafischen Dialoges ausmacht. Mit den unter der Liste platzierten Buttons kann ein Update der Liste angefordert werden bzw. das Einfügen einer Spur in die aktive Sitzung initiiert werden. Erreicht wird diese On-Demand Funktionalität dadurch, dass der NC eine Instanz eines virtuellen Clients aufstartet, der wie ein realer Mitspieler der Sitzung beitritt. Statt aus einer Soundkarte generiert der virtuelle Client seine Audiopakete aus der abgespeicherten Tonspur.

Der Aufbau eines NMP-Systems mit diesem GUI Client erfordert die Teilnahme aller vier Komponenten und ist somit ein typisches Szenario des vorgestellten Grobkonzepts mit je einem zeitkritischen und -unkritischen Elementes auf Client- und Serverseite. Analog gilt, dass die Signalisierung dem in Abbildung 7.11 dargestellten Protokollablauf entspricht.

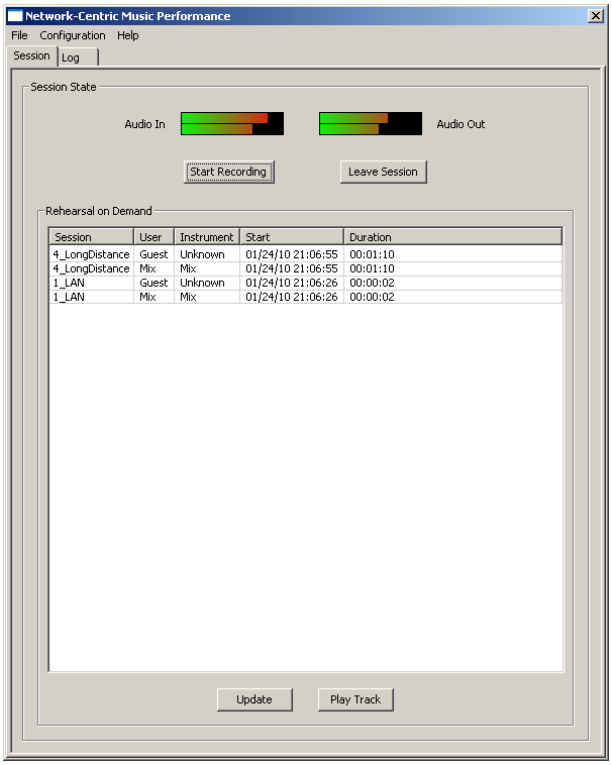

Abbildung 8.3: Graphische Benutzerschnittstelle für den NMP-Client

8.2.3 Webbasierender Dienst

Im Rahmen eines Drittmittelprojektes haben wir als Machbarkeitsstudie eine NMP Variante entwickelt, die in ein bestehendes Web-Angebot integriert wurde. Ziel dieser externen Förderung war, das existierende Angebot zum Aufspielen und Abrufen von Audiospuren oder -Stücken über die zum damaligen Zeitpunkt sehr populäre Seite *MyVirtualBand* um das gemeinsame Musizieren über NMP zu ergänzen.

Eine Realisierung dieser Studie konnte aufgrund der Flexibilität der NMP-Software Architektur sehr einfach erreicht werden. Umgesetzt wurde diese mithilfe eines eingebetteten Web Servers, über den die Anbindung der Benutzerinteraktionen an die NMP Komponenten weitergeleitet werden.

Der prinzipielle Aufbau unterscheidet sich von den bisher beschriebenen dahin gehend, dass die Kontrolle des Clients durch einen entfernten Server erfolgt. Er sieht vor, dass auf jedem teilnehmenden Rechner beim Systemstart eine Instanz des Echtzeit-Clients RC aufgestartet wird. Der Zugang des Musikers zum NMP Dienst erfolgt zunächst über den Web-Browser, in dem alle zeitunkritischen Abläufe erfolgt. Dazu zählen primär die Authentifizierung, die Konfiguration oder das Vorbereiten und Einrichten von Sitzungen. Für die meisten Aktionen waren im bestehenden Online-Angebot bereits verschiedene Funktionen enthalten, die lediglich um NMP spezifische Verwaltung von Sitzungen erweitert werden mussten.

Der Zugriff auf NMP Echtzeit-Sitzungen wurde als eigenständige Unterkategorie des *MyVirtualBand*-Dienstes realisiert, der sich aus Konfiguration, Echtzeit-Sitzungen und Abruf gespeicherter Audiospuren zusammensetzt. In Abbildung 8.4 ist das Szenario für das Echtzeit-Musizieren im *Live Sessions*-Modus abgebildet. Der Musiker erhält einen Überblick über die für ihn zugänglichen Sitzungen und deren momentaner Status. Relevant für eine einfache Verwendung des Systems sind dabei die an jeder Sitzung bereits beigetretenen Mitglieder, die Zugangsberechtigung zur Sitzung und den jeweiligen Ersteller. Diesem können erweiterte Rechte an den Sitzungen zugeteilt werden, bspw. um die Aufzeichnung von Sitzungen zu initiieren oder ge-

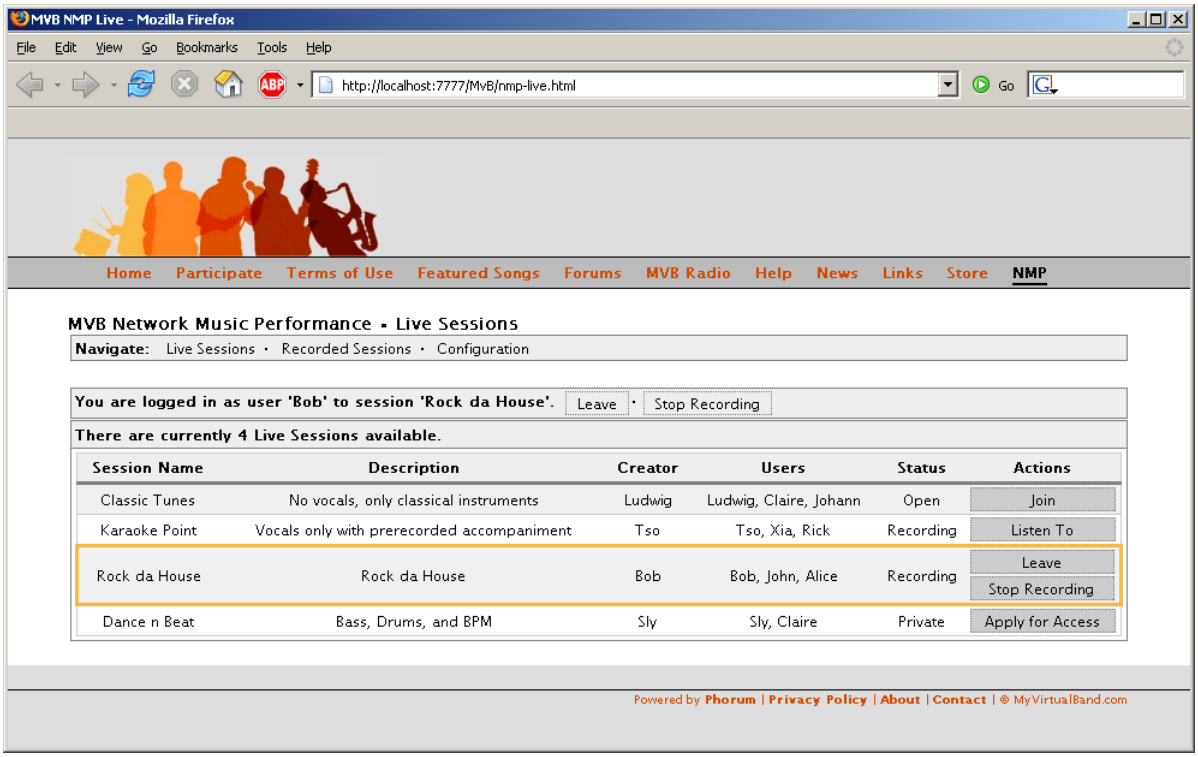

Abbildung 8.4: Echtzeit-Sitzungen im webbasierten NMP-System

speicherte Audiodaten zu löschen.

Die Interaktion mit anderen Musikern erfolgt dabei in drei Phasen:

Initiierung

Vor dem Beitritt zu einer Sitzung müssen die erforderlichen Vorbereitungen und Absprachen mit allen späteren Teilnehmern getroffen werden. Dazu können die in *MyVirtualBand*-System enthaltenen Kommunikationssysteme Chats und Foren verwendet oder gänzlich über andere Mittel vereinbart werden. Ziel dieser Vorbereitung ist das Erstellen einer NMP Sitzung und die Einrichtung erforderlicher Richtlinien. Jeder Musiker kann anschließend jederzeit an den für ihn freigegebenen Sitzungen teilnehmen. Solange er noch keiner Sitzung angehört, kann er sich über die *Listen To* Funktion als Zuhörer in eine bereits laufende Sitzung einklinken, sofern er für diese die entsprechenden Rechte besitzt.

Durchführung

Der Beitritt zu einer Sitzung erfolgt über den *Join* Button, der auf unterschiedlichen Systemebenen verschiedene Aktionen und Statuswechsel einleitet. Der Webserver übernimmt einerseits die Funktion des NS und meldet einem assoziierten RS den Beitritt des Benutzers zur Sitzung. Gleichzeitig übernimmt er auch die Rolle des NC und initiiert über die bekannten Nachrichten den Start der Audiodatenübertragung des RC auf dem Rechner des Musikers. Mit diesem Beitritt des Teilnehmers wird sein Status im System angepasst, während dessen nur noch ein stark reduzierter Funktionsumfang abrufbar ist. Als Teil der Studie wurden dabei das Verlassen der Sitzung über *Leave*, das Starten und Stoppen von Aufzeichnungen über *Start/Stop Recording* sowie das Einspielen bereits vorhandener Audiospuren (s.u.) implementiert.

Nachbearbeitung

Nach dem Verlassen einer Sitzung stehen dem Benutzer wieder alle Funktionen offen, er kann unmit-

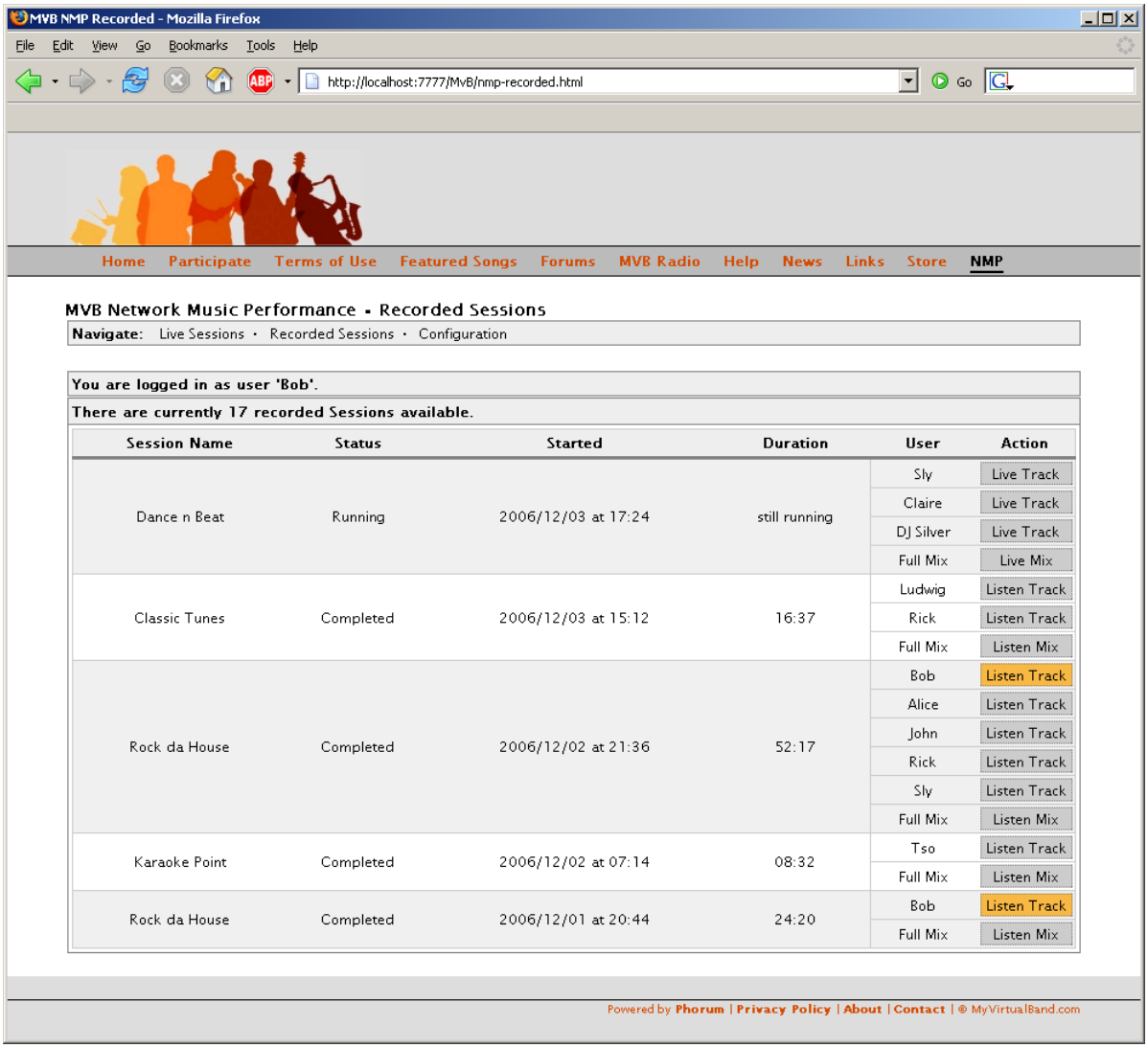

Abbildung 8.5: Web Zugang zu aufgezeichneten Sitzungen

telbar wieder einer anderen oder auch derselben Sitzung beitreten.

Die Mehrwertdienste, die sich auf Basis der aufgezeichneten Audiodaten ergeben, sind über die separate Webseite *Recorded Sessions* zugänglich. In Abbildung 8.5 ist ein Bildschirmfoto beispielhaft dargestellt. Auch hier erhält der Benutzer eine Liste der für ihr freigegebenen Sitzungen und der dabei aufgezeichneten Spuren. Anhand von Startzeitpunkt und Dauer können unterschiedliche Aufzeichnungen einer Sitzung auseinandergehalten werden. In jeder Aufzeichnung sind alle Einzelspuren und der gemischte Track vorhanden und abhängig von ihrem Status auf unterschiedliche Arten abrufbar: während die Audiospuren von zum aktuellen Zeitpunkt noch aktive Sitzungen nur über Streaming abgehört werden können, ist für bereits archivierte Spuren ein vollständiges Herunterladen der Audiodateien zur weiteren Bearbeitung möglich.

Das Streamen erfolgt im NMP Kontext direkt über die Echtzeit-Komponenten RC und RS mit virtuellen Clients. So kann das Abspielen einer Audiospur oder eines Mixes aus einer bereits aktiven Sitzung erfolgen, um fehlende Musiker zu ersetzen und somit das On-Demand Szenario zu realisieren.

Vom Standpunkt der Software Architektur und der Kommunikationspfade ähnelt dieses webbasierte NMP-System – trotz seiner augenscheinlichen Komplexität – eher dem oben in Abschnitt 8.2.1 vorgestell-

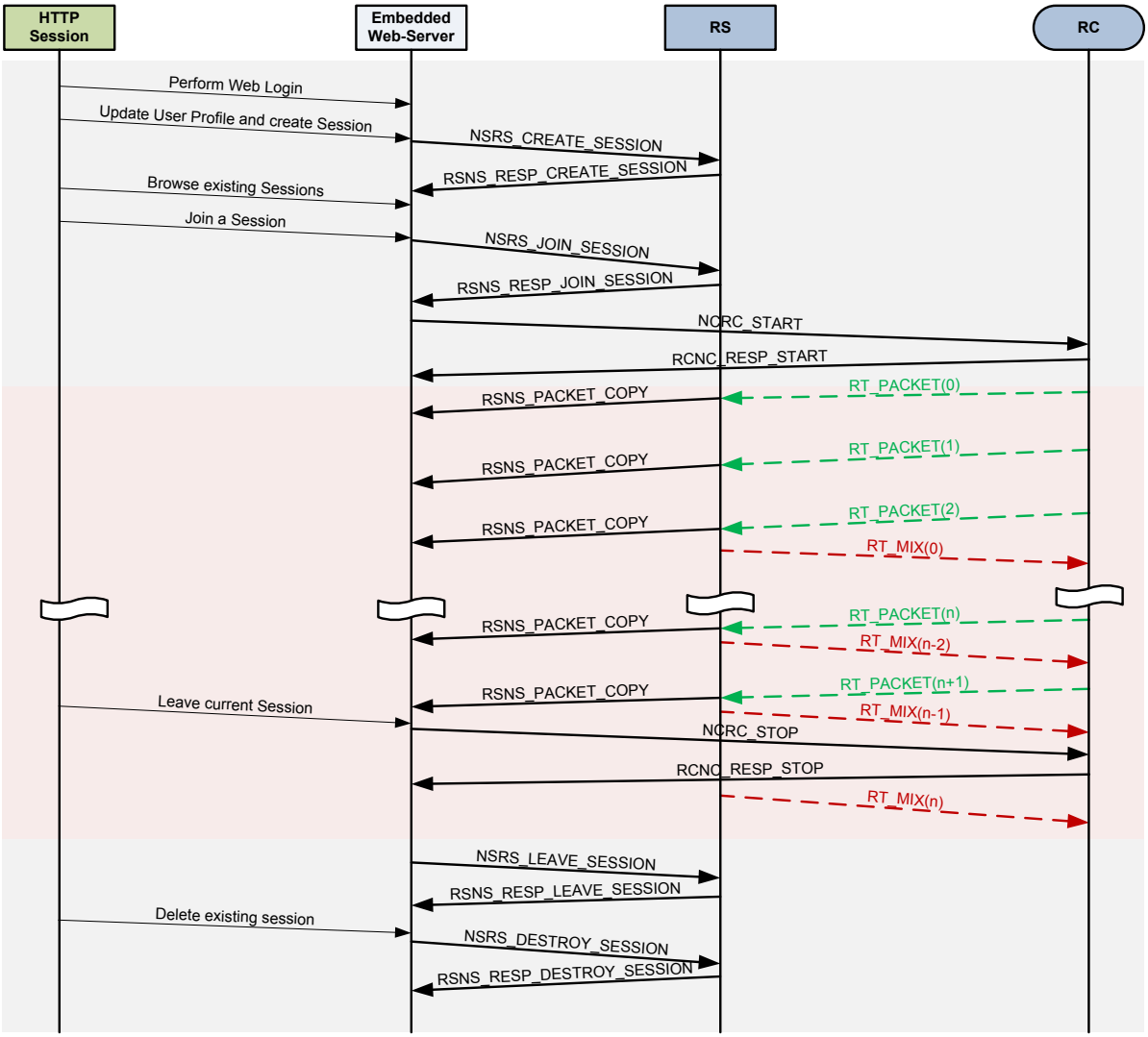

Abbildung 8.6: Server-seitige Signalisierung bei Verwendung einer Web-UI

ten Minimalsystems als der GUI Variante aus Abschnitt 8.2.2. Das wird beim Blick auf den in Diagramm 8.6 abgebildeten Protokollablauf des webbasierten System deutlich: die Funktion der in Abbildung 8.2 verwendeten Pseudo-Komponente *Debug NC/NS*, die die rudimentären Funktionen zum Aufstarten von Sitzungen enthält, sind jetzt im *Embedded Webserver* enthalten. Aus den während einer HTTP-Sitzung vom Benutzer getätigten Aktionen werden NMP relevante Eingaben gefiltert und die dazugehörigen Funktionen auf dem RS und RC initiiert.

8.2.4 Echtzeit-Client als Eingebettetes System

Der Echtzeit-Client trägt mit den in Abschnitt 5.3.1 ermittelten Faktoren wesentlich zur Latenz des NMP Gesamtsystems bei. Die dort ermittelten Einschränkungen der auf Arbeitsplatzrechnern eingesetzten Betriebssysteme bei der Verarbeitung von Echtzeitdaten erfordern einerseits eine erhöhte Latenzreserve und beinhalten ein erhöhtes Risiko, Daten nicht rechtzeitig bearbeiten zu können. Wir haben erkannt, dass diese Schwächen ein prinzipielles Problem von Multiprozess Betriebssystemen sind, auf denen die Ausführung einer Instanz des RC nur mit strikten Anforderungen an Audiohardware und -Treibern möglich ist. Trotz die-

ser Anforderungen stellt der Betrieb des RC im Arbeitsplatzrechner einen suboptimaler Kompromiss dar, da auch dann keine isochrone Datenverarbeitung gewährleistet ist.

Eine naheliegende Lösung aus diesem Dilemma ist, die RC Komponente aus den Betriebssystem spezifischen Einschränkungen heraus zu lösen und auf eine dedizierte Hardware mit einem geeigneteren OS auszulagern. Da der Funktionsumfang des RC sich auf den Austausch von Audiodaten zwischen Audio- und Netzschnittstelle beschränkt, kann diese Komponenten in Form eines eingebetteten Systems realisiert werden.

Das dazugehörige Szenario sieht dann eine Trennung des RC vom NC beim Musiker vor, wobei der RC eine kleine Box in der Nähe der Instrumente ist, die an die Audiohardware und an das Netz angebunden ist. Die Funktion des signalisierenden NC kann dabei entweder der heimische PC erfüllen, oder in einem mobilen Gerät wie PDA oder Mobiltelefon integriert werden, welche im lokalen Netz eine direkte Verbindung zum RC haben. Für die Akzeptanz beim Musiker stellt diese minimal invasive Variante die geringste Barriere für die Verwendung eines NMP-Systems dar, da die hohen Anforderungen an den RC in Form einer dedizierten Hardware kontrollierbar sind und die Anwendung von NMP Plug-and-Play fähig machen.

Ein praktischer Faktor macht die Idee des eingebetteten RC zu einem naheliegenden Ziel: nahezu alle für den Einsatz in NMP kommenden Audiokarten sind externe Komponenten, da für die Einhaltung hoher Audioqualität eine galvanische Trennung und Abschirmung von der PC-Hardware erforderlich ist. In diesen externen Gehäusen läuft die gesamte Audiodatenverarbeitung autark ab und tauscht die Audiodaten über definierte Schnittstelle mit dem angeschlossenen PC aus. Solche externen Soundkarten müssten nur leicht erweitert werden, um die Funktionalität eines RC zu erlangen und würden sich für den Musiker transparent wie seine täglich verwendeten Geräte darstellen.

Diese Idee eines eingebetteten NMP Echtzeit-Clients wurde am IBR im Rahmen einer Studienarbeit [31] untersucht. Als Plattform wurde dabei ein auf die ARM9-Architektur aufsetzendes Entwicklungsboard EDB9315A von CirrusLogic gewählt, welches über vielfältige Peripheriebausteine verfügt und dabei auch die erforderlichen Audio- und Netz-Schnittstellen bereitstellt. Als Betriebssystem wurde ein auf Echtzeitbetrieb konfiguriertes Linux verwendet, dem der Vorzug vor einem Realtime-OS gegeben wurde, weil der Hersteller alle erforderlichen Linux-Treiber bereitstellt.

Die in der Arbeit durchgeführten Untersuchungen haben die Machbarkeit eines eingebetteten NMP Echtzeit-Clients nachgewiesen. Die auf dem Entwicklungsboard verwendete ARM9-CPU mit 200ṀHz Taktfrequenz ist für alle vom RC abzuarbeitenden Funktionen mehr als ausreichend, eine Realisierung als über *PoE* (Power over Ethernet) gespeistes Endgerät ist so für einen Barriere freien Einsatz beim Musiker möglich.

Die Kapselung der kritischen und problembehafteten Anbindung an die Audiohardware in den eingebetteten RC eröffnet die allgemeine zuverlässige Verwendung eines NMP-Systems, da damit alle Unsicherheiten und potentielle Fehlerquellen bei der Auswahl und Konfiguration von Audiohardware und -Schnittstellen entfallen.

8.3 Fazit

Bedingt durch die etappenweise Entwicklung unseres NMP-Projektes hat das Design und die Implementierung der Software einen agilen Entstehungsprozess mit inkrementellen Anpassungen und Verbesserungen durchlaufen. Die Architektur der SW hat auf diesem Weg beginnend bei einer auf Funktionsfähigkeit fokussierten, unter Anwendung des XP (*Extreme Programming*) realisierten, monolithischen Variante nach verschiedenen Stadien einen heute sehr reifen Stand erreicht.

Der auf Komponenten basierende Entwurf ermöglicht einen hohen Grad an Modularität gepaart mit einer vollständigen Abstraktion von verwendeter Hardware oder Betriebssystem. Für jeden relevanten Funk-

tionsblock im NMP-System sind Schnittstellen auf verschiedenen Hierarchieebenen vorgegeben, über die Anpassungen und Erweiterungen bzw. Anbindungen zusätzlicher Module realisiert werden. Diese Flexibilität ist nicht auf die Komponenten beschränkt, sondern schließt auch die Interaktion zwischen ihnen untereinander ein, so dass die entworfenen Protokolle für Signalisierung und Datenübertragung bei Bedarf durch generische Verfahren austauschbar sind.

Mit der konsequenten Implementierung des modularen Entwurfes ermöglicht die bestehende Codebasis den Aufbau von NMP-Systemen in unterschiedlichen Konfigurationen und Komplexitätsgraden. Die aus dem grundlegenden Prinzip erwachsene Aufteilung von Client und Server in jeweils zeitkritischen und -unkritischen Komponenten können dabei auf beliebige Netzknoten verteilt werden und interagieren über die vorgesehenen Protokolle.

Dieser Umstand begründet die Skalierbarkeit unseres Systems, in dem Realisierungen auf einer Skala zwischen einem vollständig auf einem Rechner laufenden Testsystem und einem auf Serverfarmen basierenden Dienst mit einer beliebigen Anzahl von Netzknoten möglich sind. Von den in der Projektlaufzeit entstandenen NMP Varianten wurden in diesem Abschnitt stellvertretend die Ausführungen vorgestellt, die für die im nächsten Abschnitt beschriebenen Evaluation eingesetzt wurden. Der Ansatz einer Client Applikation mit graphischer Benutzerschnittstelle repräsentiert dabei Szenarien, in denen das NMP-System vollständig der Kontrolle der teilnehmenden Musiker unterliegt, während die Anbindung der NMP Komponenten über ein bestehendes Web-System die Realisierbarkeit als produktiver Web-2.0 Dienst skizziert.

Zuletzt stellt die Ausführung des Echtzeit-Clients RC auf eine dedizierte eingebettete Hardware einen bemerkenswerten Ansatz dar, der wesentliche Hindernisse beim Einsatz eines NMP-Systems durch allgemeine Anwender beseitigt. Die HW- und System spezifischen Abhängigkeiten des RC und damit verbundene Unsicherheiten bezüglich der Funktionsfähigkeit bei der Ausführung auf einem PC können damit vollständig gekapselt und in einem definierten Zustand überführt werden. Die maßgeblich durch die Verarbeitung im RC induzierten Störungen der Isochronität in den Endsystemen können so minimiert und auf die nicht kontrollierbaren Schwankungen bei der Übertragung im Netz beschränkt werden.

KAPITEL 9

EVALUATION

Zum Nachweis der erreichten Ziele wurde der entwickelte Prototyp systematisch in unterschiedlichen Konfigurationen im realen Betrieb getestet. Wir konnten dabei auf die Mithilfe und den Bewertungen von unterschiedlichen Musikern zurückgreifen. Angefangen bei Hobby-Rockbands von Studenten, über Musikschülern der Braunschweiger Musikschule bis hin zu professionellen Künstlern des Braunschweiger Theaters und Berufsmusikern von Weltklasse-Format.

In diesem Abschnitt werden mehrere Setups und die ermittelten Messergebnisse diskutiert, die zu unterschiedlichen Zeiten durchgeführt wurden und auf verschiedene Entwicklungsstufen des Prototypen aufsetzten.

Im Juli 2006 wurde NMP einem kommerziellen Interessenten vorgeführt, um den Stand unserer Arbeit zu demonstrieren. Gezeigt wurden zwei Szenarien: im LAN wurde die Arbeitsweise des Systems unter bestmöglichen Netzbedingungen und so mit der geringst möglichen Latenz gezeigt. Anschließend wurde ein Aufbau innerhalb des DFNs gewählt, um die praktische Machbarkeit von NMP auch mit einer beträchtlichen Netzdistanz zu demonstrieren. Im Laufe der anschließenden Zusammenarbeit ergab sich dabei die Gelegenheit, unser System –abweichend von unserer Zielsetzung– auch von Berufsmusikern testen zu lassen.

Mit einem weiter entwickelten Prototypen wurde NMP im Juni 2007 im Rahmen der Kompetenztage Kommunikation einer breiten Öffentlichkeit präsentiert, wobei die Verwendung eines synchronen DSL-Anschlusses (SDSL) die allgemeine Einsatzfähigkeit des Systems außerhalb geschlossener Forschungsnetze nachweisen konnte.

In den folgenden Abschnitten werden diese Testaufbauten genauer dargestellt und gewonnenen Messwerte analysiert. Abschließend findet eine Bewertung und Zusammenfassung der Ergebnisse statt.

9.1 Funktionstests im Juli 2006

Bei der im Juli 2006 durchgeführten Begutachtung von NMP sollte das Erreichen der technischen Ziele und damit die Einhaltung einer minimalen Verzögerung in den Endsystemen verifiziert und die praktische Eignung des Systems für typische Musikszenarien nachgewiesen werden.

Dafür wurden zwei Setups aufgestellt, die für die Gutachter transparent beide Nachweise erbracht haben. Grundlage dafür war ein Aufbau für drei Musiker, wobei der Standort des Servers variiert wurde, um

Aufbau	Hops	Netzdistanz RT [km]	Übertragungsverzögerung RT [μs] min	mean	max	dev
LAN	1	0.1	272	310	2630	193

Tabelle 9.1: Kennwerte im LAN

zwischen einem idealen und einem realen Netz zu unterscheiden

9.1.1 LAN: alle Komponenten innerhalb eines lokalen Netzes

Die Analyse und Optimierung der in den Endsystemen auftretenden Verarbeitungsverzögerungen und -Varianzen wurde in Kapitel 5 unter Annahme eines idealen Netzwerkes durchgeführt. In der praktischen Anwendung kommt dieser Annahme ein lokales Netzwerk nahe, in dem alle Teilnehmer über eine minimale Anzahl von Netzknoten miteinander verbunden sind. Diese Bedingungen haben wir in unserem Institutsnetz, in dem jeder angeschlossene Rechner mit jedem anderen über einen Switch direkt verbunden ist.

Vor dem Einsatz des Systems haben wir jeweils die Güte der verwendeten Netzwerke bestimmt. Dabei wurden über den Zeitraum von knapp einer viertel Stunde mit einem Sender-Reflektor Paar die Übertragungsverzögerungen von NMP-typischen Paketmustern (t_φ =2.66 ms, p_s=400 Bpp) gemessen.

Die Ergebnisse dieser Messungen sind in Tabelle 9.1 aufgeführt. Sie belegen für das LAN-Szenario eine nahezu ideale Güte des Netzwerkes, in dem die Pakete für den Hin- und Rückweg zwischen zwei Rechnern durchschnittlich 310μs benötigen. Selten treten dabei Ausreißer auf, die in der Messperiode Übertragungsverzögerungen bis hinauf zu 2630μs verursachten, die Standardabweichung wurde für diesen Zeitraum auf 193μs bestimmt.

Nach unserer Daumenregel für die Dimensionierung der De-Jitter Puffer sind somit die in Client und Server enthaltenen Puffer für den Ausgleich der Übertragungsvarianzen ausreichend, das System sollte ohne zusätzliche Puffer mit der analytisch berechneten minimalen Latenz von 13.3 ms und einer geringen Paketverlustrate operieren.

Um diese Annahme praktisch zu verifizieren, haben wir das LAN-Szenario, wie in Abbildung 9.1 skizziert, im IBR Netzwerk mit drei NMP-Clients und einem NMP-Server aufgebaut.

Dieses NMP-System wurde anschließend von zwei Kompositionen von Musikern verschiedener Professionalität und Musikrichtung erprobt und bewertet. Eine Rock-Gruppe von studentischen Hobbymusikern

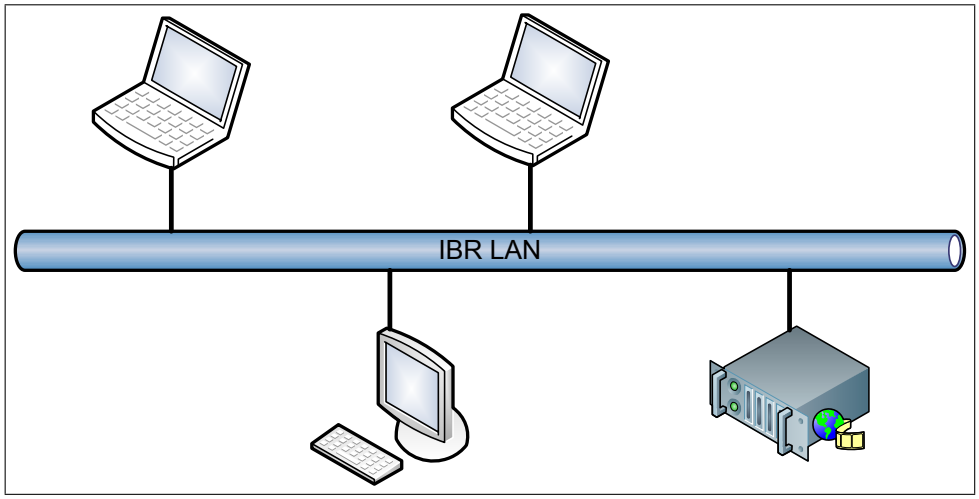

Abbildung 9.1: Aufbau des LAN Szenarios

Netzwerkpuffer			Gesamtver-	Paketverluste [%]		
Latenz	Jitter	[ms]	zögerung	mean	max	dev
0	0	0	13.3 ms	0.060	2.4	0.173

Häufigkeit von Segmenten mit verlorenen Paketen [%]											
0	1	2	3	4	5	6	7	8	9	10	> 10
67.87	22.74	5.42	1.08	1.26	0.18	0.18	0.18	0.18	0.18	0.18	0.54

Tabelle 9.2: Netzeigenschaften und Paketverlustrate im LAN_0 Szenario

(Bass, E-Gitarre und Keyboard) hat dabei einige Stücke aus dem gemeinsamen Repertoire gespielt. Die klassische Richtung wurde von drei professionellen Künstlern (E-Piano, Sopran, Tenor) des Staatstheaters Braunschweig vertreten, die ein Stück aus „La Traviata“ aufgeführt haben. An jedem Client wurden Lautsprecher Aufgestellt, so dass die Gutachter alle Stationen abgehen und sich von jeder einen Höreindruck vermitteln konnten.

Alle Sessions wurden für knapp 20 Minuten gespielt, dabei wurden von den Clients die auftretende Paketverluste protokolliert.

Alle Musiker mussten das System ohne vorheriges Einüben verwenden und waren so angehalten, wie in ihrer gewohnten Umgebung zu musizieren. Anschließend wurden sie nach ihren Eindrücken zum System befragt, wobei Benutzbarkeit, Qualitätsempfinden und die allgemeine Akzeptanz ermittelt wurden.

Die subjektive Bewertung war bei diesem Setup durchweg positiv: eine Erhöhte Latenz gegenüber der natürlichen Umgebung wurde von einigen Musikern gar nicht registriert. Diese wurde eher von den professionellen Musikern und bei Instrumenten mit kurzer Anschlagverzögerung (E-Piano) wahrgenommen, aber als nicht weiter störend empfunden.

Auch die erreichte Audioqualität wurde einstimmig als sehr gut empfunden, sowohl von den Musikern als auch von den Zuhörern. Störungen wurden über die Lautsprecher kaum registriert, nur mit hochwertigen Kopfhörern konnten vereinzelt Aussetzer wahrgenommen werden.

Zusammenfassend gaben beide Gruppen an, in diesem Szenario bezüglich der Empfindung beim Audio praktisch keinen Unterschied zum gewohnten Musizieren benennen zu können.

Diese Beurteilung wird durch die objektiven Messwerte gestützt. Während der Durchführung wurden Audiopakete, die zum Abspielzeitpunkt am Client nicht verfügbar waren, als Paketverluste registriert. Für die statistische Auswertung wurden jeweils 1000 benachbarte Pakete (entsprechend 2.66 s bei einer Paketdauer von 2.66 ms) zu Segmenten zusammengefasst und jeweils ein zusammenhängender Zeitabschnitt von mindestens 500 Segmenten betrachtet.

In Tabelle 9.2 sind die Werte dieser Messung aufgeführt und dokumentieren, dass ohne die Verwendung zusätzlicher De-Jitter Puffer eine Gesamtverzögerung von 13.3 ms erreicht werden kann, wobei die Paketverlustrate in einem akzeptablen Niveau bleibt und kaum wahrnehmbare Qualitätseinbußen verursacht.

Die aufgeführte durchschnittliche Paketverlustrate bezieht sich auf die Gesamtzahl aller versendeten Pakete; in diesem Fall konnten beispielsweise von 554000 versendeten Audiopaketen insgesamt 332 nicht rechtzeitig innerhalb von 13.3 ms wieder abgespielt werden.

Der relative Maximalwert von Paketverlusten spiegelt den schlechtesten Fall für die höchste Anzahl von verlorenen Paketen in einem Segment wieder. Ein Wert von 2.4% besagt für dieses Szenario, dass der betreffender Musiker im schlimmsten Fall damit rechnen muss, innerhalb einer Sequenz von 2.66 s insgesamt knapp 64 ms Abspielausfall hinnehmen zu müssen.

Aus der Verteilung der Paketverluste geht hervor, dass über zwei Drittel aller Segmente (67.87%) fehlerfrei sind und in weiteren 22.74% Einzelfehler detektiert werden. Mit dem Wert von 5.42% für zwei Paketverluste in einem Segment verbleiben weniger als 4% für Segmente mit drei und mehr Paketverlusten.

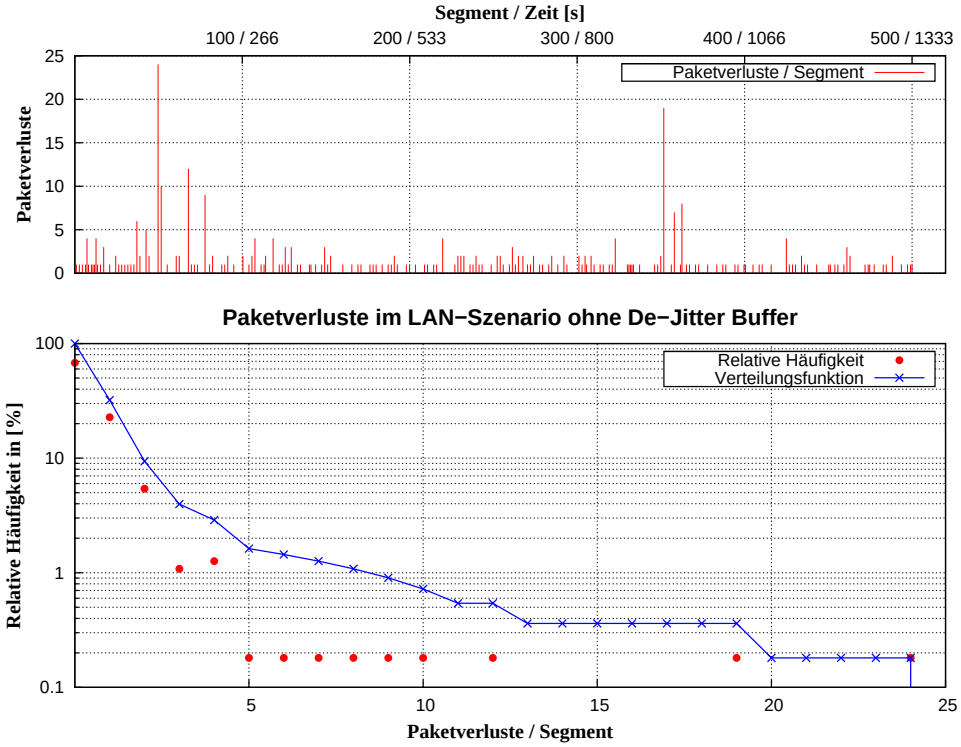

Abbildung 9.2: Paketverluste im LAN_0-Szenario ohne zusätzliche De-Jitter Puffer

Zur besseren Veranschaulichung sind die Messwerte in Abbildung 9.2 dargestellt. Im oberen Diagramm ist der zeitliche Verlauf der Paketverluste in jedem Segment aufgetragen. Es ist erkennbar, dass Paketverluste praktisch über den gesamten Zeitraum auftreten, wobei es sich meist um Einzelfehler handelt, die – wie die subjektive Bewertung bestätigt – recht effizient von den implementierten Fehlerverdeckungsverfahren versteckt werden. In den Bereichen um Sequenznummern 50 und 350 ist eine Häufung der Paketverluste sichtbar, mit einem Spitzenwert von 24 verlorenen Paketen in Sequenz Nummer 58. In diesen Segmenten steigt die Wahrscheinlichkeit für Bündelfehler, die weniger gut verdeckt werden können und daher wahrnehmbar sind.

Das Histogramm der Paketverluste pro Sequenz ist zusammen mit der inversen Verteilungsfunktion im unteren Diagramm der Abbildung aufgeführt. Die Häufigkeiten sind auf der y-Achse logarithmisch aufgetragen und verdeutlichen die hohe Dichte von Segmenten ohne bzw. mit wenig Paketverlusten.

Diese Zahlen belegen, dass die analytisch ermittelten Werte für die untere Schranke der Latenz in den Endsystemen praktisch erreicht wird und die auftretenden Paketverluste in diesem Szenario eines idealen Netzes weit unter der angepeilten Paketverlustgrenze von 1% bleiben.

Um die Paketverlustrate noch weiter herunter zu senken, wurde in einem anschließenden Szenario der De-Jitter Puffer um zwei Audiopakete erhöht. Die Gesamtlatenz erhöht sich dadurch, wie aus Tabelle 9.4 hervorgeht, um 5.3 ms auf 18.7 ms.

Die durchschnittliche Paketverlustrate sinkt mit dieser Änderung um drei Viertel auf 0.015%, allerdings verbleiben nach wie vor Segmente, in denen 1.6% der Pakete zum Abspielzeitpunkt noch nicht am Client verfügbar sind.

Die graphische Auswertung der Paketverluste in diesem Szenario ist in Abbildung 9.3 dargestellt. Unmittelbar sichtbar wird der Wegfall vieler Segmente mit nur einem Paketverlust. Über 95% aller Segmente sind jetzt fehlerfrei, die Wahrscheinlichkeit für einen Paketverlust fällt auf 0.015%. Nach wie vor existieren jedoch

Szenario	Teilnehmer	Benutzbarkeit	Audioqualität	Akzeptanz
Klassik, professionelle Musiker des Staatstheaters Braunschweig				
LAN_0	Alle	⊕⊕	⊕	⊕⊕
LAN_2	Sänger	⊕⊕	⊕⊕	⊕⊕
	Pianist	⊕	⊕⊕	⊕⊕
Rock, studentische Hobbyband				
LAN_0	Alle	⊕⊕	⊕	⊕⊕
LAN_2	E-Bass/Gitarre	⊕	⊕⊕	⊕⊕
	Pianist	⊕	⊕⊕	⊕

Tabelle 9.3: Subjektive Bewertung des NMP-Systems im LAN_0– und LAN_2–Szenario

Ausreißer von bis zu 16 verlorenen Paketen pro Segment.

Auch hierbei zeigt der zeitliche Verlauf eine Häufung von Paketverlusten an mehreren Stellen an. Bei dieser Messung sind diese bei den Sequenzen um 80, 250 und 480 erkennbar, wobei die Sequenz 252 mit 16 Paketverlusten den maximalen Verlust in dieser Messreihe aufweist. Die Verteilungsfunktion der Paketverluste pro Sequenz zeigt eine höhere Dichte in Richtung geringem Fehlerauftreten.

Auf die subjektive Bewertung des Systems hat diese Änderung eine eher nachteilige Auswirkung, wie aus der Tabelle 9.3 hervorgeht. Die Reduzierung des Anteils von Segmenten mit wenig Paketverlusten führt im Wesentlichen zu einer Entfernung von Einzelpaketfehlern. Dadurch sinkt zwar auch die Wahrscheinlichkeit von Bündelfehlern, jedoch verbleibt deren absolute Anzahl pro Zeiteinheit gleich. So konnten auch mit der höheren Anzahl von Puffern ähnlich oft Störungen wahrgenommen werden, die nicht mehr so häufig durchgeführte Fehlerverdeckung war nur schwer wahrnehmbar. Der Gewinn an Audioqualität ist mit dieser Änderung praktisch nicht relevant.

Die Bewertung der Benutzbarkeit fällt durch die Erhöhung der Latenz erwartungsgemäß negativer aus. Den Musikern, die auch anfangs die geringstmögliche Verzögerung nicht wahrnehmen konnten, fiel es schwer, überhaupt einen Unterschied auszumachen. Die anderen bemerkten den Unterschied hingegen sofort. Zwar konnten sie das NMP-System auch weiterhin ohne Einschränkungen verwenden, allerdings gaben sie an, die vorherigen Einstellungen des Systems vorzuziehen.

9.1.2 WAN: Komponenten verteilt innerhalb eines Weitverkehrsnetzes (DFN)

Nach dem Nachweis der generellen Funktionalität von NMP im lokalen Netz haben wir anschließend die Einsatzfähigkeit des Systems innerhalb des angepeilten Einsatzradius in einer typischen Konfiguration in einem Weitverkehrsnetz (WAN) demonstriert.

Wir haben dabei auf die Unterstützung unserer Kollegen vom *itm* (Institut für Telematik) an der Uni Lübeck zurückgegriffen und einen NMP-Server für diesen Test im itm LAN aufgestellt. Die Position der Clients am IBR wurde nicht verändert, hier verlief der Datenaustausch untereinander über den NMP-Server in Lübeck.

Netzwerkpuffer Latenz	Netzwerkpuffer Jitter	Netzwerkpuffer [ms]	Gesamtverzögerung	Paketverluste [%] mean	Paketverluste [%] max	Paketverluste [%] dev
0	2	5.3	18.7 ms	0.015	1.6	0.101

Segmente mit verlorenen Paketen [%]											
0	1	2	3	4	5	6	7	8	9	10	> 10
95.88	1.73	0.13	0.80	0.40	0.40	0.13	0	0	0	0.40	0.14

Tabelle 9.4: Netzeigenschaften und Paketverlustrate im LAN_2-Szenario

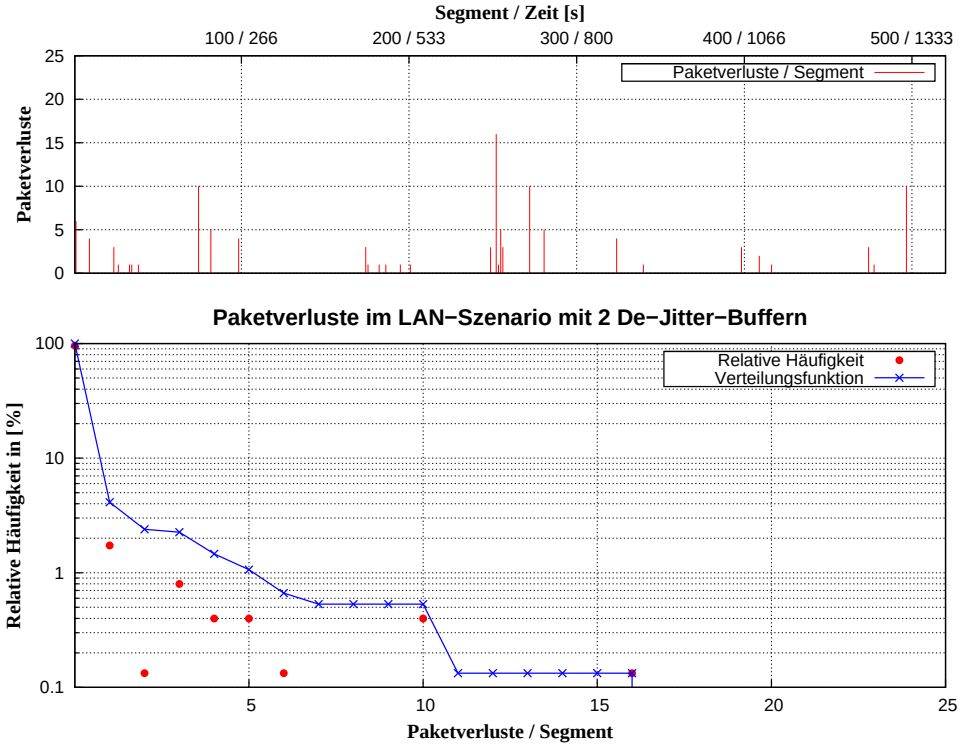

Abbildung 9.3: Paketverluste im LAN-Szenario mit zwei De-Jitter Puffern

Abbildung 9.4 skizziert diesen Aufbau. Die beiden lokalen Netze am IBR und am itm sind innerhalb des Wissenschaftsnetzes DFN über dem X-Win Basisnetz miteinander verbunden. Jeder NMP-Client ist dabei 11 Hops vom Server entfernt, wobei die Route von Braunschweig nach Lübeck über Hannover, Bremen und Hamburg führt. Diese Strecke beträgt laut Routenplaner knapp 350 km, die Netzdistanz für beide Richtungen lässt sich somit grob auf 700 km abschätzen.

In Tabelle 9.5 sind diese Netzeigenschaften und die Ergebnisse der Übertragungsverzögerungen aus den oben beschriebenen Sender-Reflektor Messungen aufgetragen. Die mittlere Übertragungsverzögerung von 9.3 ms liegt am oberen Ende des für die Netzlatenz verfügbaren Zeitbudgets. Der große Wertebereich zwischen 8.8 und 15.1 ms weist auf eine breite Streuung der Werte hin, was von dem ermittelten Wert 0.8 ms für

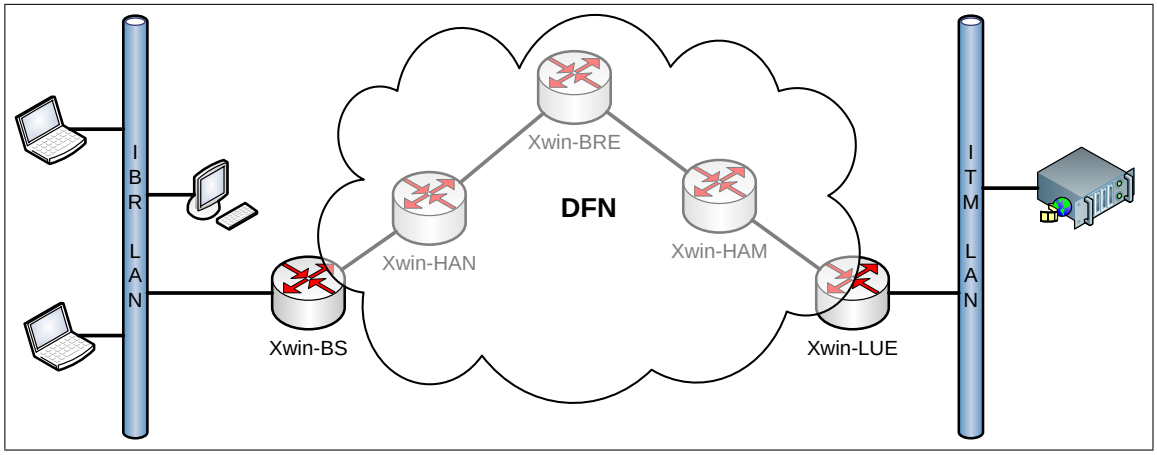

Abbildung 9.4: Aufbau des WAN Szenarios

Aufbau	Hops	Netzdistanz RT [km]	Übertragungsverzögerung RT [μs] min	mean	max	dev
WAN	11	700	8825	9315	15064	794

Tabelle 9.5: Kennwerte im WAN Szenario

den Jitter bestätigt wird.

Diese Werte erhöhen die Abspielverzögerung um vier Audiopakete (entsprechend 10.7 ms) und erfordern mindestens je einen zusätzlichen De-Jitter Puffer am Client und Server.

Mit dieser minimalen Anzahl von Netzpuffern wurde das NMP-System anschließend zwischen Braunschweig und Lübeck aufgesetzt. Die resultierende Gesamtlatenz von 29.3 ms lag bereits nur knapp unter der angepeilten Funktionsgrenze von 30 ms, so dass die Musiker hierbei diese Grenze verifizieren konnten. Beide Musikgruppen haben dabei die im LAN gespielten Stücke wiederholt.

Die Betriebsparameter dieses Setups und die gemessenen Paketverluste sind in Tabelle 9.6 aufgeführt. Im Vergleich zum LAN_0 Szenario liegen die Werte für die Paketverluste hierbei um eine Größenordnung höher. Der durchschnittliche Wert steigt auf 0.69%, der Maximalwert von 6.9% und die Standardabweichung von 1.25% belegen die hohe Streuung.

Ein Blick auf die Verteilung der Paketverluste je Segment belegt, dass mit der eingestellten Anzahl von De-Jitter Puffern die Schwankung der Übertragungsverzögerungen nicht ausreichend kompensiert werden konnte. So sind lediglich knapp 37% aller Segmente fehlerfrei, während über ein Drittel mehr als drei Paketverluste haben und über ein Sechstel mehr als 10.

Erst ein Blick auf die zeitliche Verteilung der Paketverluste in Abbildung 9.5 offenbart die Ursache für diese hohen Ausfälle. In periodischen Abständen in der Größenordnung von etwa 30 Sekunden treten Häufungen auf, die zu einem Verlust von 30 bis 40 Paketen im betreffenden Segment führen. Der Ursprung dieses Verhaltens lässt sich nicht identifizieren, da dieser innerhalb des Netzes auftritt und diesbezügliche Information nicht verfügbar sind. Plausible Möglichkeiten können periodisch ablaufende Vorgänge einschließen, die das Netz für einen kurzen Zeitraum stark belasten und in den beteiligten Komponenten Stausituationen verursachen.

Neben diesem regelmäßigen Verlustmuster sind zufällige Häufungen mit teilweise hohen Ausschlägen erkennbar, wie in den Bereichen bei Segment 270 oder 430 mit Werten von über 40 bis hinauf zu 69 Paketverlusten je Segment. Häufungen mit geringeren Werten bis hinauf zu 20 sind praktisch durchgängig enthalten. Diese Charakteristiken sind direkt aus dem Histogramm und der Verteilungsfunktion dieses Setups im unteren Diagramm der Abbildung erkennbar. Der nur zögerliche Abfall der Häufung spiegelt die Existenz vieler Segmente mit bis zu 24 Paketverlusten wieder. Auch die besondere Abweichung durch die periodischen Aussetzer gehen aus dem Histogramm anhand der erhöhten Werte zwischen 35 und 41 deutlich hervor.

Der inversen Verteilungsfunktion in diesem Graphen kann entnommen werden, dass über 10% aller Segmente mehr als 33 verlorene Pakete aufweisen. Die Wahrscheinlichkeit von Bündelfehlern, die unzureichend

Netzpuffer Latenz	Jitter	[ms]	Gesamtverzögerung	Paketverluste [%] mean	max	dev
4	2	16.0 ms	29.3	0.689	6.9	1.254

Häufigkeit von Segmenten mit verlorenen Paketen [%]											
0	1	2	3	4	5	6	7	8	9	10	> 10
37.46	10.35	12.49	6.19	4.50	3.37	3.26	1.57	1.35	0.67	1.12	17.67

Tabelle 9.6: Pufferdimensionierung und Paketverlustrate im WAN-2 Szenario

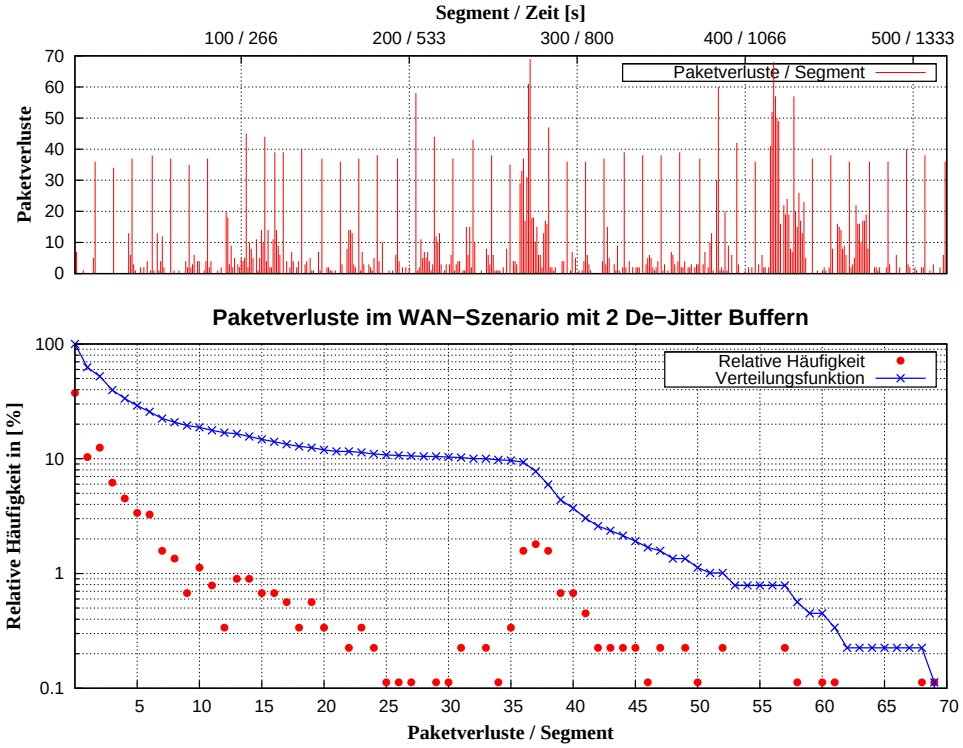

Abbildung 9.5: Paketverluste im WAN-Szenario mit zwei De-Jitter Puffern

verdeckt werden können, ist hoch.

Diese Tatsache macht sich auch beim subjektiven Qualitätsempfinden negativ bemerkbar. Diskontinuitäten bei der Audioausgabe sind hierbei durchgehend wahrnehmbar, auch über die trägen Lautsprecher. Für das Musizieren waren diese Aussetzer zwar störend, für das Spiel jedoch nicht weiter hinderlich. Den Zuhörern konnte der aus dem LAN-Szenario gewonnene Eindruck einer perfekten Musikwiedergabe durch diese Störungen hingegen nicht mehr vermittelt werden. Die regelmäßig auftretenden Knackser erinnerten eher an eine Vinyl Schallplatte als an ein digitales Musikmedium.

Der Umgang der Musiker mit der erhöhten Latenz deckt sich recht genau mit unseren Erwartungen. Diese war nun von allen Musikern deutlich wahrnehmbar. Zwar konnten alle immer noch wie gewohnt musizieren und bestätigten, keine Behinderung zu verspüren. Alle waren sich jedoch einig, dass eine weitere Erhöhung der Systemverzögerung das Zusammenspiel problematisch oder das System ganz unbedienbar machen würde.

Diese Vermutung haben wir anschließend mit einer entsprechenden Parametrisierung des WAN-Szenarios überprüft. Wir haben die Anzahl der De-Jitter Puffer in den Endsystemen um vier Pakete erhöht, die Gesamtverzögerung stieg damit auf 40 ms. Tabelle 9.7 enthält diese Parameter zusammen mit den Statistiken und dem Histogramm der gemessenen Paketverluste.

Die Statistiken belegen eine deutliche Verringerung verlorener Pakete durch das Hinzufügen der zusätzlichen De-Jitter Puffer. Die durchschnittliche Anzahl fällt um zwei Drittel auf 0.22%, Maximalverlust und Standardabweichung verbessern sich um die Hälfte auf 3.8 bzw. 0.52%.

Der Anteil der fehlerfrei übertragenen Segmente steigt auf 62.68%, zusammen mit den für den Höreindruck weniger relevanten Segmenten mit bis zu drei Paketverlusten steigt er auf über 87% gegenüber 66% bei den vorherigen Einstellungen. Der Anteil der Segmente mit mehr als zehn Paketverlusten und damit hoher Wahrscheinlichkeit von Bündelfehlern ist von über 17% auf unter 7% gefallen.

Netzpuffer			Gesamtverzögerung	Paketverluste [%]		
Latenz	Jitter	[ms]		mean	max	dev
4	6	26.7 ms	40.0	0.216	3.8	0.519

Häufigkeit von Segmenten mit verlorenen Paketen [%]											
0	1	2	3	4	5	6	7	8	9	10	> 10
62.68	10.73	7.80	5.85	1.71	0.73	0.49	1.46	0.24	0.98	0.49	6.84

Tabelle 9.7: Pufferdimensionierung und Paketverlustrate im WAN_6-Szenario

Die zeitliche Verteilung der Paketverluste im ersten Diagramm der Abbildung 9.6 gibt eine genauere Erklärung für diese Verbesserungen der Werte. Mit den zusätzlichen Puffern konnten alle vormals periodisch aufgetretenen Paketverluste kompensiert werden.

In diesem Diagramm ist weiterhin zu erkennen, dass während der ersten Hälfte der Session der Audiodatenstrom nahezu komplett fehlerfrei abgespielt werden konnte. Erst danach verschlechtert sich die Netzwerkverbindung allmählich und führt etwa ab Segment Nummer 260 zu vermehrten Auftreten von Einzelpaketverlusten, die sich bis in Größenordnungen von zehn Fehlern pro Segment ausdehnen. Kurz vor Segment Nummer 400 tritt eine weitere Erhöhung des Jitters auf, die die Verlustrate bis hinauf zu 38 Paketverlusten treibt. Dieser hohe Jitter hält in einem Abschnitt von 30 Segmenten (entsprechend 80 s) an und tritt gegen Ende der Messung ab Segment Nummer 470 erneut auf.

Die Abbildung des Histogramms im unteren Teil der Abbildung 9.6 dokumentiert den Wegfall der Segmente mit hohen Paketverlusten und die allgemeine Verjüngung des Häufigkeitsspektrums. Die Verteilungsfunktion fällt zwar deutlich steiler ab, belegt aber auch, dass mehr als zwei Prozent aller Segmente über 20 verlorene Pakete haben. Der Jitter bei diesem Netzaufbau ist also auch mit dieser zusätzlichen Anzahl von De-Jitter Puffern nicht vollständig kompensierbar.

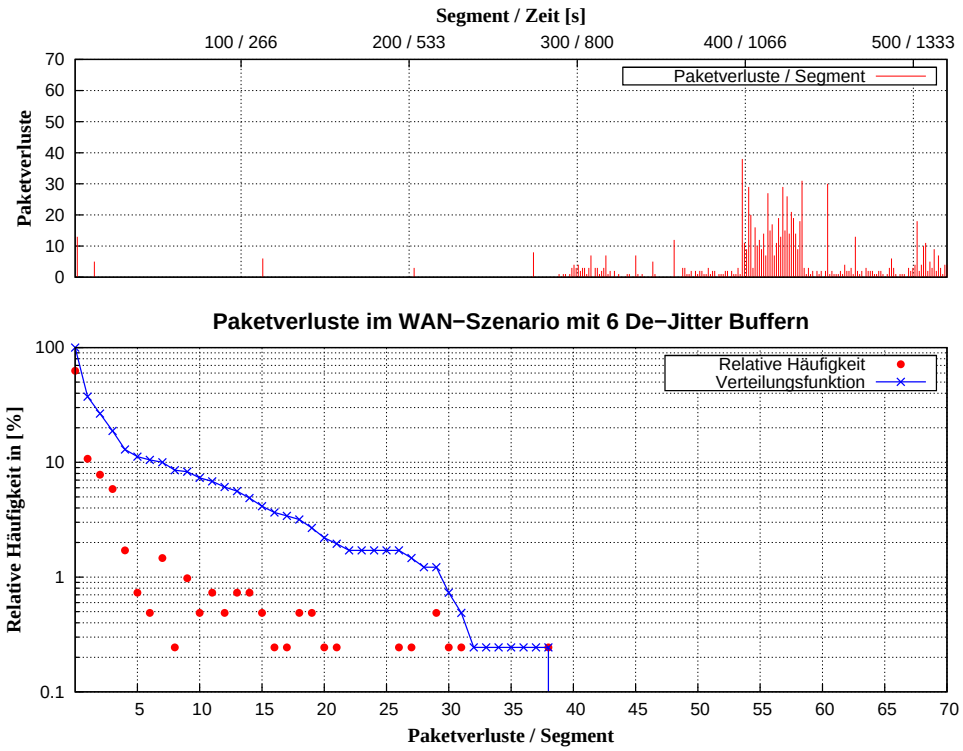

Abbildung 9.6: Paketverluste im WAN_6-Szenario

Szenario	Teilnehmer	Benutzbarkeit	Audioqualität	Akzeptanz
Klassik, professionelle Musiker des Staatstheaters Braunschweig				
WAN_2	Sänger	⊕	⊖	○
	Pianist	⊕	⊖	⊕
WAN_6	Sänger	○	⊕	⊕
	Pianist	⊖	⊕	○
Rock, studentische Hobbyband				
WAN_2	E-Bass/Gitarre	⊕	○	⊕
	Pianist	⊕	○	○
WAN_6	E-Bass/Gitarre	○	⊕	○
	Pianist	⊖	⊕	⊖

Tabelle 9.8: Subjektive Bewertung des NMP-Systems im WAN_2– und WAN_6–Szenario

Der Wegfall der periodisch auftretenden Häufung von Paketverlusten und deren durchgehende Verringerung der mittleren Höhe wurden subjektiv deutlich wahrgenommen. Die Audioqualität wurde von den Zuhörern durchweg als besser empfunden, Aussetzer und Knackser waren nur vereinzelt wahrnehmbar. Die Musiker haben diese Verbesserung nur beiläufig registriert und positiv in die Bewertung einfließen lassen, da sie mit der Kompensation der höheren Systemlatenz höher gefordert und teilweise überfordert waren.

Alle teilnehmenden Musiker gaben an, dass die hohe Verzögerung das Musizieren erschwert bzw. verhindert hat, in gewohnter Weise zu spielen. Die Mitglieder der Hobby-Band hatten hörbare Schwierigkeiten, das Tempo durchzuhalten und wurden im Verlauf des Spiels eigenen Angaben nach immer langsamer. Als sinnvolle Strategie, NMP auch in einer solchen Umgebung produktiv einzusetzen, wurde die Ergänzung eines akustischen Taktgebers (Metronom) angeregt.

Von den Sängern und der Pianistin des Braunschweiger Staatstheaters wurden diese Probleme und Schwierigkeiten bestätigt. Allerdings gelang es diesen professionellen Musiker eher, die höhere Latenz zu kompensieren. Eigenen Angaben nach trat nach ein bis zwei Minuten eine Gewöhnung ein, wobei sich die Sänger dem Tempo der Pianistin angepasst haben.

Ob diese bessere Adaptionsfähigkeit allein der Professionalität oder Erfahrung geschuldet ist, konnte im Rahmen dieser Arbeit nicht eingehend untersucht werden. Als mögliche weitere Gründe für die Unterschiede wurden Vermutungen diskutiert, dass der Gesang generell unkritischer gegenüber der Latenz sein könnte oder die Tatsache, dass beim Gesangstrio die Pianistin wie gewohnt die Rolle des Taktgebers übernahm.

Solche Fragestellungen können nur in interdisziplinärer Forschung von Musikern und Psychologen umfassend geklärt werden, für die wir mit unserem NMP-System ein geeignetes Werkzeug bereitgestellt haben.

9.2 NMP mit Berufsmusikern

Die im vorherigen Abschnitt durchgeführte Begutachtung unseres Systems durch den kommerziellen Interessenten hat zu einer Zusammenarbeit geführt, innerhalb derer wir NMP zu einem Messe Prototypen entwickelt haben, der in das bestehende Web Angebot des Interessenten eingebunden werden konnte.

Dieser Prototyp wurde im Januar 2007 auf der *IAJE* (International Association for Jazz Education) in New York und während der *NAMM 2007* (National Association of Music Merchants) Show in Anaheim demonstriert. Bei diesen Ausstellungen hat sich die Gelegenheit ergeben, das System von zwei professionellen Berufsmusikern testen und bewerten zu lassen.

Jeff Tamelier ist ein Gitarrist, der seit 1982 in verschiedenen Bands gespielt hat, darunter von 1996 bis 2006 bei *Tower of Power*. Heute ist er Präsident von *House of Hansen Productions, LLC*, einem Unternehmen, das sich mit kollaborativer Online Musik beschäftigt.

John 'JR' Robinson wird als der 'meist aufgezeichnete Drummer in der Geschichte' bezeichnet und spielte in unzählige Filmmusiken und für zahlreiche Musikgrößen. Zuletzt begleitete er 2006 Barbara Streisand als Perkussionist auf ihre Tournee.

Während der beiden Ausstellungen wurde an den Tagungsorten ein NMP Setup mit einem Server und drei NMP-Clients in einem lokalen Netz aufgebaut. Da alle Rechner direkt über einen Switch miteinander verbunden waren, ähnelten die Netzeigenschaften denen des LAN-Szenarios im IBR in Tabelle 9.1. Wir haben daher bei diesem Test keine Messungen durchgeführt und die Paketverluste nicht untersucht, sondern uns auf die subjektive Bewertung und Akzeptanz der Systemlatenz konzentriert. Diese konnten dabei im quasi idealen lokalen Netz durch die Parametrisierung der zu verwendenden De-Jitter Puffer variiert werden, so dass Verzögerungen in einem Bereich von knapp unter 32 ms mit sieben bis hinunter zu 13.3 ms ohne zusätzliche Puffer untersucht wurden.

Die Tests haben wir in zwei Abschnitte gegliedert: zunächst sollte jeder Musiker einzeln die Verzögerung wahrnehmen und bewerten, anschließend wurde das gemeinsame Musizieren geprobt und die Akzeptanzschwelle für die Systemlatenz ermittelt. Neben der Variation der Pufferlänge wurde als zusätzlicher Systemparameter das Zuschalten des Audio-Loopbacks verwendet, wobei das Eingangsaudiosignal unmittelbar dem Ausgangssignal latenzfrei hinzugemischt wurde. Dieses ermöglicht auf der einen Seite ein direktes Feedback der Eingabe, führt andererseits jedoch nach dem Mischen mit dem vom NMP-Server zurückgesandten Audiodatenstrom zu einem Versatz, der ab einer bestimmten Schwelle als Echo wahrnehmbar ist.

Während der Testdurchführung wurden die Musiker nicht über die Einstellungen der Puffergrößen informiert und bei jeder Parametrisierung nach ihrer Akzeptanz befragt. So konnte abschließend die tatsächliche Genauigkeit der Verzögerungswahrnehmung zuverlässig bestimmt werden. Die Ergebnisse dieser Auswertung sind in Abbildung 9.7 dargestellt. Auf der Abszisse ist die Systemlatenz aufgetragen, Messungen wurden jeweils bei einem Vielfachen der Paketdauer von 2.66 ms durchgeführt. Die Ordinate enthält die subjektive Bewertung des Systems in Abstufungen zwischen *unspielbar* und *transparent*, wobei die höchste Bewertung bedeutet, dass mit dem NMP-System ein Musizieren wie in natürlicher Umgebung möglich ist. Aufgetragen sind Kurven für beide Musiker jeweils mit und ohne Loopback, sowie für das gemeinsame Musizieren.

Bei Jeff Tamelier verlief der erste Teil des Tests ohne Loopback bei variabler Verzögerung recht positiv, da er mit allen Latenzen bis 24 ms sehr gut zurechtkam und auch mit darüber hinaus gehenden Latenzen spielen konnte. Der entsprechende Verlauf der Bewertungskurve zeigt diese hohe Akzeptanz: mit den beiden geringst möglichen Verzögerung von 13.3 und 16 ms wurde das System als transparent bewertet. Die Zufriedenheit fällt linear bis zu 24 ms, bei der das System immer noch als ohne vorherige Übung spielbar bewertet wurde. Mit weiter steigender Latenz fällt die Zufriedenheit rapider ab, ab 29.3 ms war es nur noch schwer bespielbar.

Für John Robinson war das erste Setup sehr problematisch: für ihn war nur mit der minimalen Verzögerung ein sicheres Spiel praktisch möglich. Bereits bei einer Latenz von 16 ms musste er sich vor dem Spiel an das System gewöhnen, mit noch höheren Verzögerungen kam er gar nicht zurecht. Das Fehlen der auditiven Antwort in der Wahrnehmung bereitete ihm dabei massive Schwierigkeiten bei der Einhaltung des Tempos. John gab an, dass diese möglicherweise nach längerem Training reduziert werden könnten, das System ihm aber auch dann kein natürliches Spiel ermöglichen würde. Hier lag offenbar bereits die Akzeptanzgrenze, denn schon mit einem zusätzlichen Puffer wurde NMP mit einer Latenz von über 16 ms für JR nicht mehr benutzbar.

Anschließend wurden die Tests mit dazugeschaltetem Audio-Loopback wiederholt, um den Musikern eine unmittelbare auditive Antwort auf ihr Spiel zu geben. Beim Perkussionisten führte dies zu einer signifikant besseren Bewertung des Systems, der zur bestmöglichen Akzeptanz bei den Latenzen von 13.3 und 16 ms führte. Bei diesem zeitlichen Versatz zwischen unmittelbar ausgegebenen und vom NMP reflektierten Audiodatenströmen war das Echo offenbar unterhalb der Wahrnehmungsschwelle bzw. wurde nicht negativ

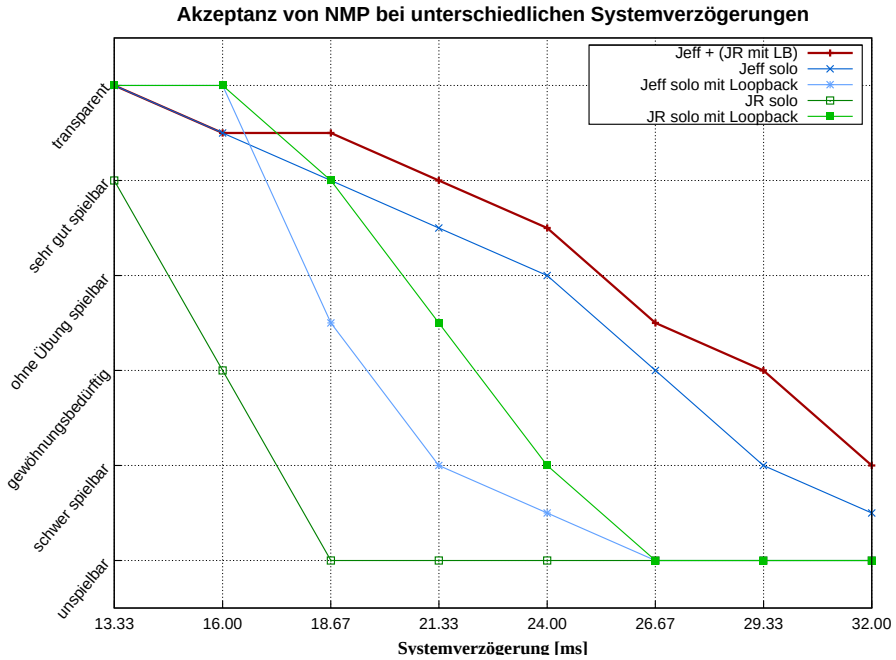

Abbildung 9.7: Akzeptanz von NMP bei Profi-Musikern in Abhängigkeit von der Latenz

empfunden, da bei Schlaginstrumenten auch in natürlicher Umgebung durch Reflexionen wahrgenommene Echos auftreten. Eine weitere Erhöhung der Latenz verstärkt den Versatz und damit die Wahrnehmung eines störenden Halls. Die anfangs gute Bewertung nimmt mit jedem zusätzlich verwendeten Puffer gravierend ab erreicht bereits mit einer Verzögerung von 24 ms die Akzeptanzschwelle.

Für Jeff zeigte die Beimischung des Audio-Loopbacks einen gänzlich anderen Effekt. Zwar nahm auch er das Echo bis zu einer Latenz von 16 ms nicht bewusst wahr und bewertete daher das System bei dieser Parametrisierung mit der höchsten Note – genau wie ohne Loopback. Doch bereits bei der ersten zusätzlichen Erhöhung der Verzögerung fiel die Beurteilung signifikant, mit einem zusätzlichen Puffer war ein Spielen der E-Gitarre nur schwer möglich. Ein zusätzlicher Puffer machte das System mit den resultierenden Latenz von 24 ms unbedienbar. Jeff gab an, dass er das Echo so deutlich wahrnahm, dass ein koordiniertes Spiel nur einige Takte lang gelang. Möglicherweise ließe sich der Effekt mildern, indem man die Pegel der verzögerten und unverzögerten Audiodatenströme variiert. Eine solche Untersuchung lag jedoch nicht im Fokus unserer Forschung und konnte auch wegen zeitlicher Einschränkungen nicht systematisch durchgeführt werden.

Stattdessen werteten wir diese Erkenntnisse als Zwischenergebnisse, die die Parametrisierung des NMP-Systems für das gemeinsame Musizieren bestimmten: John Robinsons NMP-Client wurde mit Audio-Loopback konfiguriert, Jeff Tameliers ohne. Anschließend wurden einige Musikstücke bei variierender Latenz geprobt, die Parametrisierung wurde den Musikern nicht bekannt gegeben.

Aus den Bewertungen beider Teilnehmer für jede Verzögerungsstufe wurde das arithmetische Mittel gebildet, der Verlauf ist in Abbildung 9.7 hervorgehoben dargestellt. Bemerkenswert an den Ergebnissen des gemeinsamen Spiels war, dass das NMP-System dabei besser angenommen wurde als bei den vorherigen Solo Durchgängen. Aus o.g. Gründen konnten wir diesen Effekt nicht sicher zuordnen. Am naheliegendsten schienen uns die Erklärungen, dass entweder der Wahrnehmungsfokus in den beiden Modi sich unterscheidet, oder aber die positive Überraschung, dass gemeinsames Musizieren praktisch möglich ist, in die Bewertung eingeflossen ist.

Im Wesentlichen bestätigen diese Ergebnisse die generelle Eignung von NMP auch für den professio-

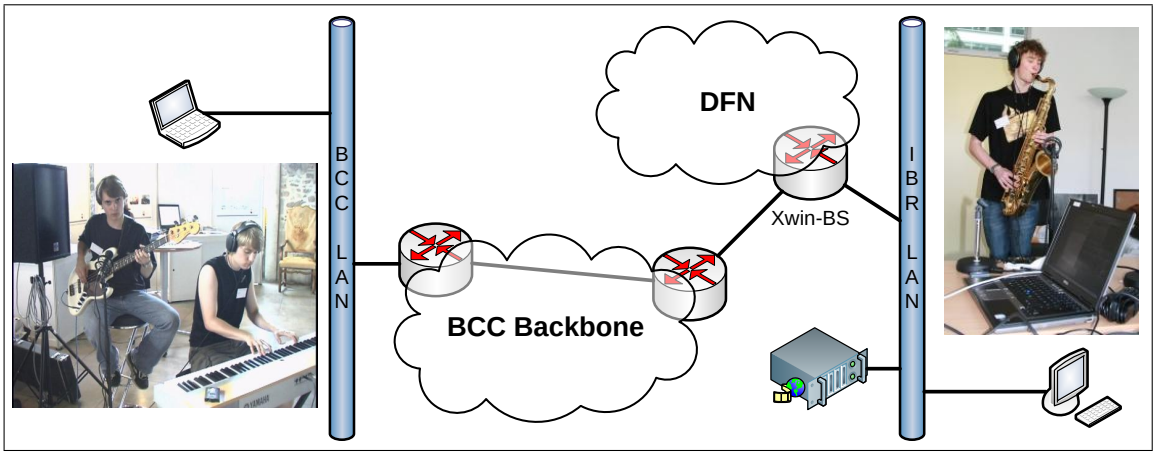

Abbildung 9.8: Aufbau des Dornse-Szenarios

nellen Bereich und für Musiker, die Studio-Bedingungen erwarten oder gewohnt sind. Allerdings offenbart es auch, dass die resultierenden Ansprüche an das System sehr hoch sind: bei den durchgeführten Tests gab es kaum Paketverluste und daher Einbußen bei der Audioqualität. Gleichzeitig wurde eine Akzeptanzschwelle von knapp 24 ms ermittelt, so dass eine zufriedenstellende Anwendung praktisch nur in lokalen Netzen gewährleistet werden kann. Als potentielle Zielgruppe für NMP im Internet scheiden Profimusiker damit aus.

In dieser Form überraschen war für uns dagegen, dass das Audio-Loopback einen derart signifikanten Einfluss auf die Akzeptanz des Systems haben kann. Hier sind sicher noch Optimierungen hinsichtlich des Pegelmischverhältnisses von gemischten und Loopback Audiodatenstrom bzw. einer zusätzlichen künstlichen Verzögerung des Loopbacks für künftige Arbeiten an NMP möglich.

9.3 NMP mit Anbindung über Synchrones DSL

Nach der im vorherigen Abschnitt beschriebenen erfolgreichen Demonstration unseres NMP Prototyps im Januar 2007 wurde eine kommerzielle Analyse der Marktfähigkeit des Systems für den USA Markt durchgeführt. Diese Untersuchung hatte zum Ergebnis, dass das Zugangsnetz in den USA für einen kommerziellen Einsatz von NMP nicht ausreichend vorbereitet war und somit eine generelle Funktionsfähigkeit von NMP für private Nutzer nicht gewährleistet werden konnte. Ohne eine kommerzielle Perspektive wurde so anschließend unsere Zusammenarbeit mit House of Hansen Ende April 2007 eingestellt. Die während dieser Kooperation entwickelten Erweiterungen und Optimierungen unseres Systems in Bezug auf Adaptivität, Robustheit und Bedienbarkeit konnte daher nicht wie vorgesehen gemeinsam evaluiert werden.

Eine Möglichkeit, unseren Prototypen der Öffentlichkeit zu präsentieren, ergab sich im Juni 2007 während der Kompetenztage Kommunikation in Braunschweig. Diese wurden im Rahmen Braunschweigs als Stadt der Wissenschaft 2007 in der Dornse des Altstadtrathauses zwischen dem 14. und 18. Juni abgehalten. Neben einer Fachtagung *Forscher Verbinden* mit Beiträgen aus den Bereichen Kommunikation und Multimedia wurde der ansässigen Forschung und Wirtschaft die Gelegenheit geboten, ihre Arbeiten zu präsentieren.

Wir konnten auf die Hilfe von drei jungen und talentierten Schülern der Musikschule Braunschweig zurückgreifen und haben bei dieser Ausstellung ein NMP Setup mit zwei Clients demonstriert. Der NMP-Server und ein Client wurden innerhalb unseres Instituts, ein weiterer NMP-Client vor Ort aufgestellt. Im IBR hat Jonte Köster Saxophon gespielt, während auf der Ausstellung Justus Bode am E-Piano und Niklas Tikwe am Bass musizierten. Die beiden Instrumente vor Ort wurden erst analog gemischt und dann mit dem NMP-Client verbunden, am IBR war das Saxophon direkt über ein Studiomikrofon mit dem Client verbunden.

Aufbau	Hops	Netzdistanz RT [km]	Übertragungsverzögerung RT [μs] min	mean	max	dev
SDSL	5	20	4760	5217	8703	518

Tabelle 9.9: Netzeigenschaften über SDSL zwischen Dornse und IBR

Der Aufbau dieses Setups ist in Abbildung 9.8 skizziert. Die Netzanbindung am Veranstaltungsort wurde vom lokalen Internet Provider BCC bereitgestellt und basierte auf einen breitbandigen synchronen DSL (SDSL) Zugang, der innerhalb des BCC Backbones QoS unterstützt. Für unseren Stand wurde eine Bandbreite von 2 Mbps reserviert, so dass die Übertragung nicht durch gleichzeitige Netzbelastungen von anderen Ständen gestört wurde, an denen in vielen Fällen Internet-Telefonie oder andere Online-Anwendungen demonstriert wurden. Zusätzlich wurde unsere Verbindung als latenzkritische VoIP-Anwendung markiert, so dass unsere Daten innerhalb des BCC Netzes mit höherer Priorität behandelt wurden als konkurrierender Verkehr durch Surfen oder Downloads. So konnte einerseits ausgeschlossen werden, dass bei Stauungen im BCC-Netz unsere Pakete verworfen wurden, andererseits war sichergestellt, dass Verzögerungen nur durch konkurrierenden VoIP-Verkehr verursacht werden konnte.

Die Durchführung wurde begünstigt dadurch, dass BCC einen Übergangsknoten zum DFN in Braunschweig unterhält, so dass der NMP-Client in der Dornse mit dem NMP-Server im IBR über fünf Hops verbunden war. Die Netzdistanz in beide Richtungen betrug schätzungsweise 20 km, so dass dieser Aufbau ein NMP-Szenario repräsentiert, bei dem alle Teilnehmer und der Server innerhalb einer Großstadt aufgestellt sind und über eine vergleichbare Netzanbindung verfügen.

Die gemessenen Paketlaufzeiten sind in Tabelle 9.9 aufgeführt und belegen die gute Eignung dieser Verbindung für NMP. Die minimale Laufzeit von 4.7 ms war höher als die für die angenommene Netzdistanz erwartete, ließ sich jedoch auf die erhöhte Verarbeitungszeit der QoS-Parameter unserer Pakete in den Routern des BCC Backbones zurückführen. Der höchste Ausschlag über den mittleren Wert von 5.2 ms ging bis hinauf zu 8.7 ms, die Standardabweichung lag bei 518μs. Die Güte der Verbindung lag damit zwischen der nahezu idealen im lokalen Netz des IBR und der zwischen Braunschweig und Lübeck über das DFN.

Anders als bei den Messungen im Juli 2006 war unser Prototyp bei dieser Messung in der Lage, sich dynamisch an die gegenwärtigen Netzeigenschaften anzupassen und unter anderem die Anzahl der De-Jitter Puffer im Betrieb zu variieren. Für die Entscheidungsfindung werden dabei kontinuierlich umfangreiche Statistiken der Paketverlustwahrscheinlichkeiten für die gerade gewählte Anzahl von Puffern sowie möglicher Alternativen berechnet und ausgewertet. So ist es nicht mehr erforderlich, für die Ermittlung der Statistiken bei unterschiedlichen Einstellungen verschiedene Messungen durchzuführen. Stattdessen ermöglichen die für die Adaption der Pufferlängen gesammelten Messdaten nachträglich eine Aussage darüber zu treffen,

Netzwerkpuffer Latenz	Jitter	[ms]	Gesamtverzögerung	Paketverluste [%] mean	max	dev
2	0	5.3	18.7	0.260	3.9	0.492
2	2	10.7	24.0	0.034	0.8	0.095
2	4	16.0	29.3	0.005	0.1	0.023

Latenz [ms]	Häufigkeit von Segmenten mit verlorenen Paketen [%] 0	1	2	3	4	5	6	7	8	9	10	> 10
18.7	50	9.8	9	7	8	3.4	2.6	1	0.8	1.4	1.2	5.8
24.0	80.4	13.6	2.6	0.8	1.2	0.8	0.2	0	0.4	0	0	0
29.3	94.8	5.2	0	0	0	0	0	0	0	0	0	0

Tabelle 9.10: Netzeigenschaften und Paketverlustrate im SDSL Szenario

Teilnehmer	Benutzbarkeit	Audioqualität	Akzeptanz
Jazz, Schüler der Musikschule Braunschweig			
E-Piano	⊕	⊕⊕	⊕
Bass	⊕⊕	⊕	⊕
Saxophon	⊕⊕	⊕	⊕⊕

Tabelle 9.11: Subjektive Bewertung des NMP-Systems im $SDSL_2$–Szenario

welche Paketverluste bei einer bestimmten Pufferanzahl aufgetreten wären.

Während der Demonstration wurde die Dimensionierung der Pufferlänge im NMP-System so parametrisiert, dass über einen Zeitraum von 10 s nicht mehr als 0.1% an Paketverlust toleriert werden soll. Bereits nach wenigen Sekunden ging das System in einen stabilen Zustand über, der während der gesamten Vorführung nicht verändert wurde. Dabei wurden zusätzlich zu den zwei Puffern für die Netzverzögerung zwei weitere für die Jitter Kompensation verwendet. Die Gesamtverzögerung von 24 ms war damit deutlich unter den angestrebten Wert von 30 ms. Gleichzeitig war, wie aus Tabelle 9.10 ersichtlich, die mittlere Paketverlustrate von 0.034% sehr gering.

Die Verteilung der Paketverluste bei dieser Einstellung verdeutlicht, dass die Wahrscheinlichkeit für Mehrfachfehler sehr gering war: 80.4% aller Segmente waren fehlerfrei, mit den weiteren Anteilen von 13.6 und 2.6% für einen bzw. zwei Paketverlusten wurden bereits über 96.6% aller Segmente abgedeckt. Die größten Ausreißer bildeten Segmente mit bis zu acht Paketverlusten. Wahrgenommen wurden diese Fehler jedoch praktisch nicht, die Musiker und auch die Zuhörer bewerteten die gebotene Audioqualität als sehr gut.

Die Musiker bewerteten das System darüber hinaus insgesamt als sehr gut verwendbar, sie konnten ohne vorheriges Proben direkt ihre Session abhalten. Die Latenz wurde als nicht weiter störend empfunden, was sicherlich zum Teil auf die gewählte Musikrichtung Jazz zurückzuführen ist, der von Improvisation lebt und eine, im Vergleich zu anderen Stilen, höhere Latenztoleranz hat.

Bei Bedarf wäre die Gesamtverzögerung in diesem Szenario um zwei Puffer und damit 5.33 ms reduzierbar, ohne die Audioqualität signifikant zu verschlechtern. Dabei wäre zwar die durchschnittliche Paketverlustrate auf 0.26% gestiegen, die Audioqualität bliebe – auch bei der dabei zu erwartenden maximalen Verlustrate von 39 Paketverlusten pro Segment – akzeptabel. Ein Blick auf die Verteilung der Paketverluste verdeutlicht, dass der für die Wahrnehmung kritische Anteil von Segmenten mit hoher Paketverlustanzahl relativ niedrig ist: nur 10% aller Segmente haben mehr als sechs Fehlstellen. Verglichen mit dem im Juli 2006 durchgeführten Messungen im DFN-WAN wäre dies ein für den NMP Einsatz deutlich besserer Aufbau.

Auch in anderer Richtung bietet dieser Aufbau eine Reserve für die Variation der Pufferlänge. Sind die Musiker bereit und in der Lage, mit einer Gesamtverzögerung von 29.3 ms zu arbeiten, können zwei zusätzliche Puffer für das De-Jittering verwendet und damit Paketverluste praktisch vollständig verhindert werden. Dann können nämlich 94.8% aller Segmente völlig fehlerfrei abgespielt werden, den übrigen 5.2% fehlen lediglich je ein Paket. Mehrfachverluste wären dabei höchst unwahrscheinlich, die Einzelfehler wären durch die implementierten Verdeckungsmaßnahmen nur schwer wahrnehmbar.

Abbildung 9.9 stellt für dieses Szenario die gemessenen Paketverluste als zeitliches Auftreten und als Histogramm dar. Der zeitliche Verlauf im oberen Diagramm stellt die Anzahl von nicht ausgespielten Paketen pro Segment bei einer Verwendung von zwei, vier oder sechs Puffern für die Netzverzögerung überlappend dar. Es sind Bereiche sichtbar, die bei der Verwendung eines geringen Puffers zu deutlichen Ausschlägen führen, wie in Segment 437 bis hinauf zu 39 Verlusten. Zwei zusätzliche Puffer für das De-Jittering verringern die Höhe der Ausschläge deutlich auf ein Niveau, das keine wahrnehmbaren Störungen der Audioausgabe gewährleistet. Die verbleibenden Fehlstellen bei der Verwendung zweier zusätzlicher Puffer sind als Einzelfehler erkennbar. Die Betrachtung individueller Werte belegt, dass keine Vorhersage darüber getroffen werden

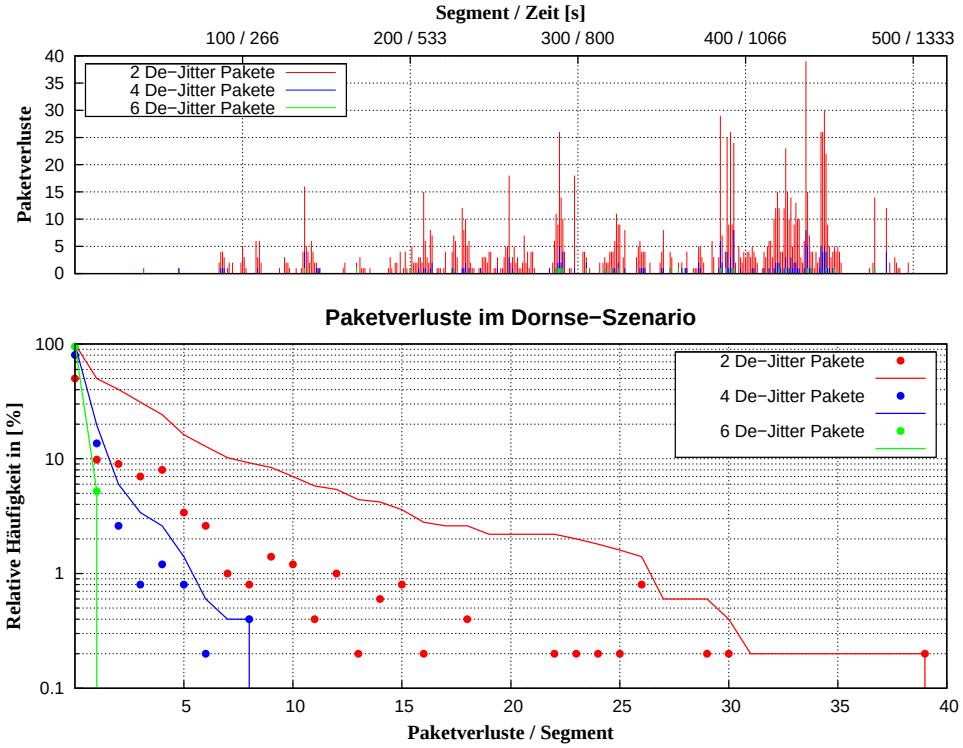

Abbildung 9.9: Paketverluste im Dornse-Szenario

kann, wie sich die Verwendung zusätzliche Puffer auf die Fehleranzahl pro Segment auswirkt. So reduzieren sich die Paketverluste in Segment 478 bei Verwendung von zwei zusätzlichen De-Jitter Puffern von 14 auf nur noch eine Fehlstelle, während kurze Zeit später bei Segment 485 von den 12 Paketverlusten vier verbleiben.

Die Histogramme und die inverse Verteilungsfunktionen für die drei Parametrisierungen im unteren Teil der Abbildung 9.9 verdeutlichen sehr anschaulich, wie sich das Spektrum der Paketverluste pro Segment Richtung Fehlerfreiheit verjüngt. Der dem Musiker gewährte Kompromiss zwischen Systemverzögerung und Audioqualität wird hiermit direkt sichtbar.

9.4 Zusammenfassung und Diskussion der Ergebnisse

Die in diesem Abschnitt beschriebenen Live Demonstrationen weisen alle an unser NMP-System gestellten Funktionalitäten und Eigenschaften nach. Dazu gehört zunächst einmal der fundamentale Nachweis, dass verteiltes und Netz gestütztes Musizieren überhaupt praktikabel ist. Wir haben darüber hinaus gesehen, dass NMP in stabilen Netzen von den meisten Musikern sehr gut akzeptiert wird – gleichzeitig wurden wir aber auch mit den hohen Anforderungen von Berufsmusikern konfrontiert, die das System eher skeptisch aufgenommen haben. Auf technischer Seite haben wir gezeigt, dass die analytisch bestimmte minimale Latenz von 13.3 ms unter idealen Netzbedingungen auch praktisch eingehalten wird. Die Wahl der verwendeten Netze belegt, dass NMP aus dem lokalen Netz eines Labors in reale Netze bewegt werden kann, ohne wesentlich die Anwendbarkeit zu behindern.

Diese Ergebnisse werden im Folgenden zusammenfassend diskutiert.

9.4.1 Grundlegende Praktikabilität von NMP

Die vielleicht wichtigste Erkenntnis aus diesen Untersuchungen ist die, dass das NMP-System von allen Musikern ohne vorhergehende Einarbeitung verwendet werden konnte und als Werkzeug gut akzeptiert wurde. Dabei bestätigte sich unsere Annahme, dass die Akzeptanz am ehesten bei Amateuren und Fortgeschrittenen zu suchen ist. Beim Freizeitmusiker ist diese dadurch gegeben, dass aufgrund eines weniger stark ausgeprägten Gespürs für Verzögerung und Audioqualität die Toleranz dafür recht hoch ist. Fortgeschrittenen Teilnehmern fallen die Einschränkungen zwar deutlich eher auf, mit ihrer Erfahrung und Professionalität haben sie jedoch weniger Schwierigkeiten, diese zu kompensieren.

Eher unerwartet waren die positiven Ergebnisse bei Berufsmusikern. Diese geben bei Befragungen vorab gewöhnlich an, keinerlei zusätzliche Latenz tolerieren zu können. Tatsächlich konnten wir jedoch ein Setup aufsetzen, in dem sie netzgestützt musizieren konnten. Da für die Zuhilfenahme des Audio-Loopbacks eine individuelle Parametrisierung des Systems erforderlich war, und gleichzeitig eine Akzeptanz von NMP nur mit in lokalen Netzen erreichbaren Verzögerungen gegeben war, kann eine allgemeine Praktikabilität von NMP in realen Netzen im Profisegment nicht gewährleistet werden. Dieses wurde von dem beiden teilnehmenden Musikern auch so bestätigt: eigenen Einschätzung nach wäre das System für Jam-Sessions verwendbar, nicht jedoch im produktiven Einsatz.

Eine andere positive Erkenntnis war, dass die von uns angestrengte und einzuhaltende maximale Verzögerung von 30 ms sich als weniger strikte Anforderung erwiesen hat. Wie bei den im Juli 2006 durchgeführten Tests im WAN_6-Szenario ermittelt, konnten die Musiker auch mit Verzögerungen von 40 ms arbeiten, die Akzeptanz und Bedienbarkeit des Systems variierte dabei in Abhängigkeit von Musikrichtung und Professionalität der Musiker. Den Angaben der Musiker aus dem Staatstheater Braunschweig zufolge konnte die erhöhte Latenz bereits nach wenigen Minuten Übung gut kompensiert werden. Für einen praktischen Einsatz des NMP-Systems birgt diese Aussage das Potenzial, die Systemverzögerung bei entsprechender Eingewöhnung zu erhöhen, und damit den Anwendungsradius von NMP signifikant erweitern zu können.

9.4.2 Eignung unterschiedlicher Netze für NMP

Der Funktionstest von NMP innerhalb der gewählten Netzwerken mit den in Tabelle 9.12 aufgeführten Kennwerten hat nachgewiesen, dass NMP in unterschiedlichen Netzen einsetzbar ist, solange die erforderliche Netzwerkkapazität bereitgestellt und das Netz keinen ungewöhnlich starken Jitter hat.

Ein lokales Netz, bei dem die Anzahl aktiver Netzkomponenten klein ist – idealerweise ist jeder Rechner mit jedem anderen direkt über einen Switch verbunden – stellt die optimale Umgebung für NMP dar, da die Übertragungsverzögerung und der Jitter im Netz verschwindend gering sind. Sinnvolle Szenarien für einen solchen Einsatz sind beispielsweise Campus-Netze, die unterschiedliche Institute miteinander verbinden, private lokale Netze in Studentenwohnheimen oder ganzen Ortschaften, wie auch kommerzielle Netze in großen Firmen, Hotels und Ferienanlagen. Tabelle 9.13 belegt, dass überall hier NMP bei einer Verwendung von zwei zusätzlichen De-Jitter Puffern mit einer sehr kurzen Systemverzögerung von unter 20 ms eingesetzt werden kann und dabei eine sehr hohe Audioqualität gewährleistet, wie die Paketverlustrate im verwendeten LAN von knapp 0.1% belegt.

Das WAN-Szenario innerhalb des DFNs zwischen Braunschweig und Lübeck repräsentiert alle Aufbauten innerhalb breitbandiger und stabiler Netze bei relativ großer Netzdistanz. Die zu überbrückende Entfernung in beiden Richtungen liegt mit 700 km am oberen Ende des theoretischen Einsatzradius von NMP, gleiches gilt für die Anzahl der zu passierenden Netzkomponenten von 11 Hops. Viele Verbindungen zwischen zwei Knoten im DFN sind meist über weniger Zwischenstationen (meist sechs bis acht Hops) erreichbar, der hier gemessene Jitter wird üblicherweise nicht erreicht. In diesem Szenario konnte die Parametrisierung so

Aufbau	Hops	Netzdistanz RT [km]	Übertragungsverzögerung RT [μs] min	mean	max	dev
LAN	1	0.1	272	310	2630	193
WAN	11	700	8825	9315	15064	794
SDSL	5	20	4760	5217	8703	518

Tabelle 9.12: Kennwerte in den verwendeten Netzwerke

variiert werden, dass mit zwei De-Jitter Puffern die Gesamtverzögerung knapp unter der anvisierten Grenze von 30 ms gehalten werden konnte wenn eine Paketverlustrate von 0.7% toleriert werden kann. Langsamere Musik oder eine höhere Latenztoleranz von Musikern erlauben eine Erhöhung der Pufferanzahl zugunsten besserer Audioqualität – wir haben Messungen mit vier zusätzlichen Puffern durchgeführt, wodurch die Gesamtverzögerung auf 40 ms erhöht und die Verlustrate im Gegenzug auf 0.22% gedrückt wurde.

Diese Messergebnisse belegen, dass NMP innerhalb des DFNs das gemeinsame Musizieren praktisch Deutschland weit ermöglichen kann, wenn der Server innerhalb der Netztopologie zentral aufgestellt wird. Zu einem beispielsweise in Frankfurt platzierter NMP-Server lägen die Netzdistanzen zu jedem beliebigen Rechner im DFN auch im ungünstigsten Fall nur unwesentlich über den hier gemessenen, gleiches gilt für die Anzahl der Hops. Netzgestütztes Musizieren im Rahmen von Kooperationen zwischen verschiedenen Hochschulen in Deutschland sind genau so möglich wie virtuelle Bühnen zwischen den Bewohnern von Studentenwohnheimen, sofern sie breitbandig am DFN angebunden sind. Nach vorherigem Training oder bei weniger latenzkritischen Musikstücken lässt sich der Einsatzradius auch über Deutschland hinaus vergrößern, so dass NMP-Projekten wie der ECMA [50] mit Musikern aus Polen, der Tschechien und Frankreich eine virtuelle Bühne für den gemeinsamen Musikunterricht bieten könnte. Gerade der Bereich E-Learning im breitbandigen Forschungsnetzen birgt ein hohes Potenzial für NMP.

Zuletzt haben die erfolgreich durchgeführten Demonstrationen über ein SDSL- Zugangsnetz belegt, dass NMP nicht auf Forschungsnetze wie dem DFN beschränkt ist, sondern auch für den Privatanwender mit einem entsprechenden Zugang praktisch verfügbar wäre.

Im vorgeführten Aufbau konnte aufgrund der geringen Varianz der Übertragungsverzögerung auf zusätzlicher De-Jitter Puffer verzichtet werden und so eine Gesamtverzögerung von 18.7 ms und eine durchschnittliche Paketverlustrate von 0.26% eingehalten werden. Die resultierende Audioqualität ist zufriedenstellend, so dass das verbleibende Budget an verfügbarer Verzögerung für die Überbrückung größerer Netzdistanzen verwendet werden könnte. Auch hier wäre das gemeinsame Musizieren deutschlandweit möglich, wenn der Server zentral platziert wird und alle Clients über ein vergleichbares Zugangsnetz verfügen.

Für eine kommerzielle Nutzung von NMP müsste ein Anbieter neben der Bereitstellung der SDSL-Zugänge auch eine größere Anzahl von NMP-Servern betreiben, die idealerweise in Ballungsräumen liegen. Unter der Annahme, dass Musiker zum netzverteilten Musizieren eher dann zusammen kommen, wenn sie

Aufbau	Netzwerkpuffer Latenz	Jitter	[ms]	Gesamtverzögerung [ms]	Paketverluste [%] mean	max	dev
LAN_0	0	0	0	13.3	0.060	2.4	0.173
LAN_2	0	2	5.3	18.7	0.015	1.6	0.101
WAN_2	4	2	16.0	29.3	0.689	6.9	1.254
WAN_6	4	6	26.7	40.0	0.216	3.8	0.519
$SDSL_0$	2	0	5.3	18.7	0.260	3.9	0.492
$SDSL_2$	2	2	10.7	24.0	0.034	0.8	0.095
$SDSL_4$	2	4	16.0	29.3	0.005	0.1	0.023

Tabelle 9.13: Zusammenhang von Wartezeit und Paketverlusten in Live Demonstrationen

auch im realen Leben gemeinsam spielen und so auch geografisch nicht weit auseinander liegen, werden Sessions wahrscheinlicher, bei denen alle Teilnehmer innerhalb eines Ballungszentrums liegen und so einen NMP-Server in unmittelbarer netztopologischer Nähe nutzen können. In diesem Fall könnten vergleichbare Netzeigenschaften wie die hier gemessenen erwartet werden, die Musiker könnten das System also in einem Bereich zwischen akzeptabler Qualität bei einer Gesamtverzögerung von unter 20 ms und nahezu perfekter Audioqualität bei einer Verzögerung von unter 30 ms parametrisieren.

9.4.3 Parametrisierung und Abwägung zwischen Audioqualität und Latenz

In den beschriebenen Tests haben wir verschiedene Musikrichtungen berücksichtigt, die von Musikern unterschiedlicher Professionalität gespielt wurden. Wir konnten für jede Session mindestens ein Setup finden, in dem die Musiker ohne Training und mit hoher Akzeptanz spielen konnten.

Wie effektiv eine Abwägung zwischen Audioqualität und Systemverzögerung durch die Parametrisierung der Anzahl von De-Jitter Puffern möglich ist, wurde besonders im WAN-Szenario deutlich. Innerhalb des DFNs lagen die Netzeigenschaften zwischen Braunschweig und Lübeck bereits an der Grenze des Machbaren. Die Verzögerung in den Endsystemen zusammen mit der Netzlatenz und der erforderlichen Jitter-Kompensation lagen im WAN_2-Szenario nahe an der Toleranzgrenze von 30 ms, gleichzeitig traten in periodischen Abständen Störungen im Netz auf, die zu einer Vielzahl von Paketverlusten führten. Erst eine deutliche Erhöhung der De-Jitter Puffer konnte im WAN_6 einen Großteil der Paketverluste eliminieren, im Gegenzug musste dafür eine Latenz von 40 ms toleriert werden.

Tabelle 9.14 fasst die subjektiven Bewertungen der Musiker für diese beiden Szenarien zusammen und belegt, dass wir im Rahmen dieser Messungen keinen systematischen Zusammenhang zwischen Parametrisierung und Bewertung der Benutzbarkeit und Akzeptanz feststellen konnten. Während die Beurteilung der erreichten Audioqualität plausibel mit den objektiven Werten für die Paketverlustrate korreliert, hängt die Bedienbarkeit des Systems von verschiedenen Faktoren ab.

Unsere Messungen waren zwar nicht umfangreich genug, um Regelmäßigkeiten aufzudecken, allerdings bestätigen sich einige naheliegende Annahmen. Musiker an Instrumenten mit kurzer Anschlagverzögerung wie beim E-Piano erwarten ein unmittelbares auditives Feedback der motorischen Aktion und reagieren empfindlicher auf höhere Latenzen als beispielsweise bei Saiten-Instrumenten oder Gesang, die eine längere Anschlags- und Abklingdauer haben. So bestätigen die Angaben der Pianisten sowohl in der klassischen Zusammensetzung als auch in der Rock-Formation, dass sich für sie die Bedienbarkeit des Systems im WAN_6-Szenario wegen der höheren Latenz deutlich verschlechterte und damit insgesamt schlechter akzeptiert wurde als das WAN_2-Szenario. Anders herum gaben die Sänger und Saiten-Instrumentalisten an, die erhöhte Verzögerung beim WAN_6-Szenario zugunsten einer besseren Audioqualität zu tolerieren und dieses vorzuziehen.

Die Unterschiede bei der Bewertung der klassischen und der Rockmusik lassen zusätzliche Rückschlüsse auf Zusammenhänge zwischen der Beurteilung und Musikrichtung bzw. Professionalität zu. So bewerten die Freizeitmusiker die vom System gebotene Audioqualität im WAN_2-Szenario durchweg besser als die professionellen Musiker des Staatstheaters Braunschweig, was daran liegen kann, dass letztere an höherwertigere Geräte und besserer Qualität gewöhnt sind. Warum diese Profimusiker eher in der Lage waren, sich an die höhere Latenz im WAN_6 anzupassen, kann verschiedene Gründe haben: möglich, dass ihre Erfahrung sie dazu befähigt, sich an solche Gegebenheiten anzupassen; vielleicht lag es auch daran, dass die Sänger auch auf der Bühne mit größeren Abständen zueinander arbeiten und höhere Latenzen auch im realen Musizieren gewohnt sind; nicht zuletzt kann auch das im Vergleich zur Rockmusik deutlich niedrigere Tempo in der Klassik eine höhere Latenztoleranz bedingen. Aufgrund der eingeschränkten Teilnehmerzahl sind solche

Musiker	Subjektive Bewertung WAN_2/WAN_6		
	Benutzbarkeit	Audioqualität	Akzeptanz
Klassik, Staatstheater Braunschweig			
Sänger	⊕/◯	⊖/⊕	◯/⊕
Pianist	⊕/⊖		⊕/◯
Rock, studentische Hobbyband			
E-Bass/Gitarre	⊕/◯	◯/⊕	⊕/◯
Pianist	⊕/⊖		◯/⊖

Tabelle 9.14: Änderungen der subjektiven Bewertung im WAN_2/WAN_6-Szenario

Überlegungen an dieser Stelle spekulativ und wären durch eine systematische Untersuchung mit statistisch signifikanten Stichproben zu überprüfen – eine Aufgabe, die nicht im Fokus dieser Arbeit stand und deren Arbeit sprengen würde. Am naheliegendsten ist jedoch der Zusammenhang in einer Kombination dieser und möglicher zusätzlicher Faktoren zu suchen.

Ebenfalls möglich ist, dass auch eine intensive Untersuchung dieser Zusammenhänge keine abschließende objektive Kriterien festlegen kann, da diese zu sehr von subjektiven Gegebenheiten abhängen. Dies lässt sich recht deutlich an zwei Musikern klar machen, die über identische Fähigkeiten verfügen und mit dem gleichen Instrument ein gemeinsames Stück spielen. Einer wird möglicherweise eine geringere Latenz bevorzugen, weil sein schlechteres Gehör sich an die damit einhergehende geringere Audioqualität weniger stört.

Aus diesem Grund haben wir in unserem Prototypen auf eine automatische Parametrisierung verzichtet und geben dem Musiker die Freiheit, abweichend von einer vorgegebenen Standardeinstellung die Abwägung zwischen hoher Audioqualität und geringer Latenz selbst zu bestimmen.

9.5 Fazit

Die in diesem Abschnitt beschriebenen Demonstrationen belegen primär eines: innerhalb seiner Grenzen funktioniert NMP zuverlässig. Musiker sind in der Lage, ohne vorheriges Training das System sofort zu verwenden und produktiv Musik zu machen. Den hohen Anforderungen an eine geringe Latenz können existierende Netze gerecht werden, die Bandbreite an möglichen Szenarien reicht von einem Setup im lokalen Netz bei nahezu perfekter Audioqualität und einer Gesamtverzögerung unter 20 ms bis zu einem Aufbau in Weitverkehrsnetzen mit Netzdistanzen von bis zu 700 km bei akzeptabler Audioqualität und einer Gesamtlatenz von ca. 30 ms. Prinzipiell hat dabei der Anwender die Freiheit, über die Parametrisierung der Anzahl zu verwendender De-Jitter Puffer eine Abwägung zwischen Systemverzögerung und Audioqualität zu treffen.

KAPITEL 10

ZUSAMMENFASSUNG

Wir haben uns in dieser Arbeit der Problemstellung des interaktiven netzverteilten Musizierens im Internet zum heutigen Stand der Technik gewidmet und uns eingehend mit den herausfordernden Problemen und Schwierigkeiten beschäftigt. Basierend auf die vorgestellten Lösungsansätzen haben wir einen Software Prototypen entwickelt, mit dem wir die prinzipielle Machbarkeit von NMP demonstrieren können. Die Zusammenarbeit mit unterschiedlichen Teilnehmern in einem breiten Spektrum zwischen Hobby- und Berufsmusikern und die dabei erzielte positive Bewertung haben eine allgemeine Akzeptanz unseres NMP-Systems nachgewiesen.

Diese Erkenntnis ist vielleicht das signifikanteste Ergebnis der Arbeit, denn sie widerlegt unsere anfänglichen Zweifel, dass Musizieren als soziales Ereignis in einer rein auditiven virtuellen Umgebung nicht durchführbar sein könnte. Tatsächlich ist NMP für Musiker auf Anhieb und nahezu transparent benutzbar – wenn die Systemlatenz in Bereiche liegt, die Musiker auch in natürlicher Umgebung vorfinden. Für diese Anforderung haben wir empirisch einen Wert von 30 ms ermittelt und uns als Hürde auferlegt, die etwa der Verzögerung auf einer realen Bühne entspricht und sich als für alle teilnehmenden Musiker als beherrschbar erwiesen hat.

Mit dieser Anforderung haben wir im ersten Teil dieser Arbeit eine umfassende Latenzanalyse eines NMP-Systems durchgeführt und daraus Anforderungen und Einschränkungen für das Design und die Implementierung des Prototypen abgeleitet. Kernaussage der Analyse ist, dass von unserem Latenzbudget von 30 ms etwa die Hälfte als unvermeidbare Bearbeitungs- und Aufenthaltszeit in den beteiligten Endsystemen anfällt und der Rest für die Übertragungsverzögerungen im Netz verfügbar ist.

Aus diesen beschränkenden Faktoren lässt sich ein Applikationsradius für netzverteiltes Musizieren in einer Größenordnung von 500 km Netzdistanz abschätzen.

Ob eine praktische Realisierung tatsächlich in die Nähe dieser theoretischen Schranke gelangt, hängt im Wesentlichen von zwei Faktoren ab: erstens müssen Entwurf und Entwicklung des Systems strikt auf eine Minimierung der Latenz als wichtigstes Funktionskriterium ausgelegt sein, um die analytisch ermittelte Verweilzeit der Audiodaten in den Endsystemen nicht zu überschreiten. Zweitens muss das verwendete Netz derart definierten Anforderungen genügen, dass ein Betrieb innerhalb der Latenzschranke mit hoher Audioqualität durchführbar ist.

Die erste Bedingung erfordert enorme Sorgfalt und Anstrengungen bei der Umsetzung, bleibt dafür je-

doch innerhalb der technischen Vorgaben kontrollierbar. Für die Herausforderungen, die sich bei der isochronen Datenverarbeitung auf Arbeitsplatzrechnern ergeben, haben wir Lösungswege vorgestellt und praktisch umgesetzt. Mit einer selektiven Auswahl geeigneter Audiohardware und Treiberschnittstellen, sowie der systemnahen Implementierung zeitkritischer Funktionen im Interrupt Kontext können Server- und Client-Komponenten auf gewöhnlichen PCs unter den aktuellen Betriebssystemen die erforderliche minimale Latenz auch praktisch einhalten. Darüber hinaus haben wir nachgewiesen, dass sporadische, auf Arbeitsplatzrechnern nicht zu vermeidende Störungen behoben werden können, indem die zeitkritischen Komponenten auf dedizierte eingebettete Hardware ausgelagert werden. Unter diesen Idealbedingungen kann die ermittelte Verzögerung in den Endsystemen garantiert werden.

Die zweite Bedingung entzieht sich dagegen nahezu vollständig der Kontrolle durch den Benutzer, da sie mindestens eine genaue Kenntnis der Netzpfade und der Charakteristiken des verwendeten Netzes erfordert. Für den Betrieb von NMP ist beim Client durchgängig eine Übertragungskapazität von mindestens 0.5 Mbps in ein- und ausgehender Richtung erforderlich, während der Server diese Datenrate multiplikativ für jeden Client bearbeiten und übertragen muss. Mit heutiger Zugangstechnik scheint die Anbindung der Clients an ein NMP-System möglich, für den Server sind heute für den Privatanwender verfügbare Internetzugänge dagegen noch nicht ausreichend. Weitaus schwerer lässt sich die Übertragungsverzögerung in einem Netz beurteilen: wie ein zu verwendendes Netz aufgebaut ist und wie es intern funktioniert, ist praktisch immer unbekannt. Angaben über Netztopologie, wie die Anbindung des Zugangsnetzes an das Backbone ist oder wo in den Netzen von Mitbewerbern Daten ausgetauscht (gepeert) werden, sind meist nicht verfügbar oder werden als Betriebsgeheimnisse der beteiligen Provider nicht bekannt gegeben.

Eine Abschätzung der Verwendbarkeit von Netzen allein aufgrund ihrer nomineller Kennzahlen hat sich als unpraktikabel herausgestellt, da sie weder konkret genug ist noch Informationen über die zeitliche Konstanz vorliegen – geschweige denn verbindlich gewährt werden.

Tatsächlich haben sich im Projektverlauf alle für den Privatanwender verfügbaren Zugangsnetze als für den praktischen Einsatz von NMP als untauglich erwiesen. Die Kapazität der in Deutschland am häufigsten verwendeten Internetzugänge über ADSL wurde zwar bis an die technischen Grenzen ausgebaut, jedoch mit einem überwiegenden Fokus auf dem Downlink. Den 16 Mbps in dieser Richtung stehen im Uplink 1 Mbps entgegen, die häufig bis auf die Hälfte des nominellen Wertes gedrosselt sind. Darüber hinaus muss für den Betrieb von NMP die Vorwärtsfehlerkorrektur abgeschaltet (bzw. das FastPath Feature 'freigeschaltet') werden, was wiederum zu vermehrten physikalischen Übertragungsfehlern führt und den Zugang für NMP unbrauchbar macht.

Bei Zugängen über Kabel sind auf den ersten Blick bessere Voraussetzungen gegeben, die jedoch in Abhängigkeit der gleichzeitig aktiven Teilnehmerzahl stark schwanken und somit für NMP ebenso ungeeignet sind. Teilweise haben wir sogar in Hochleistungsnetzen Eigenschaften vorgefunden, die eine sinnvolle Verwendung von NMP verhindert haben. So wurden bei einigen Sitzungen innerhalb des Deutschen Forschungsnetzes (DFN) zwischen dem IBR und der Uni Lübeck periodische Unterbrechungen bei der Übertragung in der Größenordnung von 30 ms innerhalb weniger Minuten gemessen – für die meisten Anwendungen kaum erkennbar, für NMP dagegen nicht nutzbar. Die einzig praktikable Maßnahme zur Beurteilung, ob ein zu verwendendes Netz den Anforderungen von NMP genügt, ist daher das Messen.

Das für den privaten Nutzer am nächsten realisierbare NMP-Szenario haben wir innerhalb unseres Projektes mit einer symmetrischen DSL Anbindung vorgefunden. Im Rahmen einer Live Demonstration wurde eine Sitzung präsentiert, bei der ein Client über einen SDSL Zugang an den im DFN gelegene Server verbunden war. Über diese Anbindung mit einer garantierter Kapazität von 2 Mbps in beide Richtungen konnte NMP ideal betrieben werden. Allerdings sind solche Zugänge zum jetzigen Zeitpunkt nur punktuell und ausschließlich für gewerbliche Kunden verfügbar. Für NMP Privatanwender ist somit SDSL heute neben hoher

Kosten auch wegen der geringen Abdeckung ungeeignet.

In einem Flächenland wie Deutschland besteht darüber hinaus das Dreiecksproblem beim Peering: möchten zwei Musiker aus Hamburg miteinander online musizieren, die ihren Internetzugang nicht beim selben Provider haben, müssen die Daten an einem Peering Knoten zwischen den Netzen der Anbieter ausgetauscht werden. Erfolgt dies bspw. über den DE-CIX in Frankfurt, fällt bereits die Sitzung zwischen den beiden Nachbarn wegen der Enormen Netzdistanz aus dem Anwendungsradius von NMP.

Die gewonnene Einsicht, dass der Betrieb von NMP zum jetzigen Zeitpunkt eine explizite Selektion des zu verwendenden Netzes erfordert, ist auf dem Weg zur allgemeinen Anwendbarkeit des Systems nur ein temporäres Hindernis. Mit dem kontinuierlichen Ausbau bestehender Infrastrukturen und der Einführung neuer Technologien wird der technische Fortschritt in naher Zukunft die Verfügbarkeit der für NMP geeigneten Netzzugänge bedingen und unser System unmittelbar einsetzbar machen.

Nach der Klärung der prinzipiellen Umsetzbarkeit von NMP sind wir im zweiten Teil dieser Arbeit vom *'Ob'* auf das *'Wie'* eingegangen. Wir haben ausführlich die Anforderungen ermittelt, die eine Realisierung von NMP mit der gegebenen Latenzschranke begleiten und jeweils geeignete Lösungsansätze erarbeitet und vorgestellt. Die angetroffenen Schwierigkeiten waren dabei teilweise von Anfang an bekannt, während andere subtiler Natur waren und erst im Projektverlauf zu Tage getreten sind.

Die Einschränkungen beim Entwurf der Kommunikationsprotokolle, die fehlende Möglichkeit, verlorene Daten erneut zu übertragen und das damit implizit vorhandene Risiko von Diskontinuitäten im Audiodatenstrom sind dabei typische Vertreter der ersten Gruppe. Auch die Anforderungen an Design und Implementierung zur Einhaltung minimaler Verweildauer der Daten in den Endsystemen sind unmittelbar aus der Latenzanalyse hervorgegangen und in das initiale Lösungskonzept eingeflossen, welches den Aufbau unserer NMP Anwendung als verteiltes System vorsieht. In diesem System ist der zeitkritische Austausch von Audiodaten zwischen Client und Server in dedizierte Module ausgelagert, die von darüber liegenden zeitunkritischen Modulen kontrolliert werden.

Das geringe Latenzbudget verhindert die Anwendung etablierter Ansätze aus artverwandten Anwendungen und hat zu eher untypischen Lösungen geführt, wie die Wahl geeigneter Audiocodecs gezeigt hat. Für NMP lassen sich etablierte Codecs für Musik aufgrund ihrer hohen algorithmischer Verzögerungen nicht verwenden, ebenso wie verbreitete Sprachcodecs, die nicht die erforderliche Audioqualität erreichen. Unter verschiedenen Verfahren haben wir den weniger bekannten Audiokompressor WavPack als bestmögliche Wahl identifiziert und für die Verwendung mit NMP erweitert.

Für die Synchronisation multipler Audiodatenströme innerhalb einer Sitzung können die üblicherweise auf Zeitstempeln beruhende Verfahren nicht angewandt werden, da weder die erreichbare Zeitsynchronisation über NTP genau genug ist, noch die zur Synchronisation erforderlichen Datenratenanpassungen innerhalb der Latenzschranke durchführbar sind. Wir haben stattdessen ein Verfahren entwickelt, welches ohne zusätzliche Verzögerung arbeitet und die erforderlichen Operationen als Paketverwurf über die immanent vorhandenen Paketverluste ohne zusätzliche Diskontinuitäten im Audiodatenstrom erreicht.

Um die Auswirkungen von unvermeidbar vorhandenen Paketverlusten auf die Audioqualität zu verringern, sind Maßnahmen zur Fehlerverdeckung für NMP obligatorisch. Bekannte und effiziente Verfahren setzen auf ausgefeilte Mechanismen zur vorauseilenden Absicherung oder reaktiven Verdeckung von Datenlücken, erhöhen jedoch immer die Systemverzögerung aufgrund zusätzlicher Pufferung und sind für NMP nicht verwendbar. So werden bspw. Strategien für unterschiedliche Mehrfachfehler eingesetzt, während bei NMP bereits die Zeit fehlt, einen solchen zu identifizieren, bevor er effektiv zum Tragen kommt. Wir haben für dieses zentrale Problem verschiedene Ansätze untersucht und das verwendete Verfahren vorgestellt, bei dem Paketverluste während Echtzeit-Sitzungen ohne zusätzliche Verzögerung kaschiert, für den folgenden On-Demand Gebrauch dagegen fehlerfrei rekonstruiert werden.

Die Erkenntnis, dass die Auswirkung von Paketverlusten auf die wahrgenommene Audioqualität und die potentielle Verbesserung durch unterschiedliche Fehlerverdeckungsmaßnahmen höchst subjektiv ist, gehört zu der zweit genannten Kategorie von Herausforderungen. Erst die Evaluation unterschiedlicher Ansätze im laufenden Betrieb hat so offenbart, dass eine komplexe analytische Rekonstruktion einer Datenlücke gegenüber einem Auffüllen der Lücke mit Rauschen subjektiv oft keinen gravierenden Vorteil bringt. Dementsprechend beinhaltet unser vorgestelltes Verfahren zur Fehlerkompensation unterschiedliche Strategien, die an die persönlichen Vorgaben des Benutzers angepasst werden können.

Personalisierung ist auch für die Abwägung zwischen Systemlatenz und Audioqualität essenziell. Musiker an Instrumenten mit schnellem Feedback (Perkussion, E-Gitarre) können die Latenz höher als die Qualität gewichten, andere Musiker tolerieren eine höhere Verzögerung als Preis für bessere Audioqualität. Wir haben das Konzept konfigurierbarer Präferenzmatrizen entworfen, anhand derer die Systemlatenz aufgrund der laufenden Messung angepasst wird, um den Benutzer die nach seinem Vorgaben bestmögliche Erfahrung mit NMP zu bieten.

Eine weitere Funktion, die nicht vorhergesehen und erst als Ergebnis von Evaluation mit NMP hinzugefügt wurde, ist die Relevanz des lokalen Feedbacks. Während unserer Tests mit Anfängern und Fortgeschrittenen wurde die Anwendbarkeit von NMP bei einer Systemverzögerung von 30 ms durchweg als sehr gut bis transparent bewertet. Erst die Untersuchung mit Berufsmusikern, die selten auf der Bühne und praktisch immer in Studioumgebung arbeiten, hat ergeben, dass ein lokales Feedback mit minimaler Latenz für manche Musiker unverzichtbar ist. Als Konsequenz unterstützt unser NMP Prototyp einen Betriebsmodus, bei dem der lokale Audiodatenstrom mit einem personalisierbaren Latenzversatz vor dem gemischten ausgegeben wird.

Es ist abzusehen, dass bei einem intensiveren Einsatz unserer NMP Anwendung mit unterschiedlichen Musikern noch andere subjektive Faktoren zutage treten und Raum für Verbesserungen bieten, um netzverteiltes Musizieren als praktische Ergänzung zu realen Übungen und Sitzungen zu etablieren. Den Grundstein dafür haben wir mit dieser Arbeit gelegt und mit unserem Prototypen ein funktionsfähiges Werkzeug bereitgestellt, der Musikern in geeigneten Netzen eine intuitive Teilnahme an interaktiven Sitzungen auf virtueller Bühne ermöglicht.

10.1 Ausblick

Abseits des antizipierten technologischen Fortschritts beim Ausbau der Netztechnik und der damit einhergehenden allgemeinen Verwendbarkeit von NMP beim Privatanwender, sind die ermittelten Herausforderungen und verbliebenen Einschränkungen prinzipieller Natur und bieten aus unserer Sicht keinen Raum für signifikante Verbesserungen.

Die Verweilzeit der Audiodaten in den Endsystemen, die bereits im optimalen Fall knapp die Hälfte des Latenzbudgets aufzehrt, ist prinzipbedingt nicht weiter reduzierbar. Die Verarbeitung der Audiodaten erfolgt auf dem gesamten Datenpfad blockbasiert: angefangen vom Austausch zwischen Audiohardware und Betriebssystem über die gesamte Verarbeitung im Rechner bis hin zum paketbasierten Datentransport über das Netz.

In einem solchen blockbasierten Verarbeitungssystem legt die verwendete Blockgröße die algorithmische Verzögerung fest, indem sie die Pufferverzögerung definiert. Eine beliebige Reduzierung der Blockgröße wird in der Praxis durch technische Vorgaben begrenzt. In einer Audioanwendung limitiert zuallererst die verwendete Hardware diesen Wert nach unten, damit der Rechner und das verwendete Betriebssystem in der Lage bleiben, die Audiodaten in der geforderten Frequenz auszutauschen. Die in NMP verwendete Blockgrö-

ße von 128 Audiosamples mit einer Blockzeit von $t_\varphi = 2.66\,ms$ markiert momentan das untere Ende des zum jetzigen Stand der Technik Realisierbaren.

Einer darüber hinaus gehenden Verringerung der Blockgröße stehen technische und praktische Einschränkungen entgegen. Zum einen existieren für den Einsatz in PCs praktisch kaum Audiokarten, die mit kleineren Blockzeiten arbeiten, da Betriebssysteme für Arbeitsplatzrechner nicht dafür ausgelegt sind, eine derart hohe Verarbeitungsfrequenz kontinuierlich und fehlerfrei bewältigen zu können. Zum anderen wirkt sich eine kleinere Blockgröße negativ auf Datenverarbeitung und Transport aus. Audiokompressionsverfahren verlieren mit kleinen Blockgrößen ihre Effizienz, gleichzeitig sinkt das Verhältnis von Nutz- zur Datenmenge aufgrund von Anwendungs- und Transportprotokoll-Headern. So lässt sich mit dem Einsatz eingebetteter Systeme die Minimierung der Blockgröße bis hinunter auf einzelne Samples auf die Spitze treiben. In dem dann zu transportierende Datenstrom mit einer Rate von 25 Mbps sind dann weniger als 2 % Audiodaten enthalten. Gleichzeitig muss jede Komponente auf dem Datenpfad, einschließlich aller Netzknoten, in der Lage sein, die Datenpakete durchgängig und kontinuierlich in Abständen von knapp 20.8 μs zu bearbeiten.

Die von uns getroffene Wahl stellt die zum jetzigen Zeitpunkt optimale Abwägung zwischen Pufferlatenz, Effizienz und Umsetzbarkeit dar, die voraussichtlich auch durch zukünftige Entwicklungen keine wesentlichen Änderungen erfahren wird. Die Verweilzeit der Audiodaten in den Endsystemen kann somit dauerhaft als fixer Anteil an der Systemverzögerung von NMP angesehen werden.

Welche Netzdistanz sich mit dem verbliebenen Latenzbudget überbrücken lässt, ist dagegen stark von der verwendeten Netzinfrastruktur abhängig. Und hier tut sich momentan einiges: ADSL wurde bis an die technischen Grenzen ausgereizt, um als Tripple-Play Lösung als alleinige Anbindung alle Datenarten bedienen zu können, stößt jedoch mit dem aufkommenden hochauflösenden HD-Medien an seine Grenzen. Der Übergang zur nächsten Generation von Zugangstechniken mit Glasfaseranbindung bis zum Hausanschluss (*FTTH*, Fiber To The Home) steht aktuell an und wird in naher Zukunft dem Privatanwender einen Internetzugang mit den Eigenschaften lokaler Netze bieten können.

Mit Erreichen der Sättigung bei der Übertragungskapazität rückt – getrieben von interaktiven Netzwerkspielen – der Faktor Latenz immer stärker in den Fokus. Die nächste Generation der Netzzugangstechniken versprechen eine deutlich verringerte Latenz auf der Strecke zwischen Zugangsnetzen und Backbones und eine höhere Stabilität und Kontinuität des Datentransports. Gleichzeitig entschärfen die Provider das Dreiecksproblem, indem die Anzahl von Peering-Knoten erweitert wird, so dass geografische und Netzdistanzen sich annähern.

Es ist somit absehbar, dass mit dem Übergang zur nächsten Zugangstechnik die für den Betrieb von NMP erforderlichen Eigenschaften gegeben sein werden und netzgestütztes Musizieren den Weg der IP-Telefonie in den Alltag folgen wird. Mit unserem Prototypen steht für den Einstieg bereits heute eine stabile, funktionale und erweiterbare Basis zur Verfügung.

10.2 Neuere Entwicklungen

In einem derart dynamischen und mit enormen Tempo fortschreitenden Gebiet wie der Informationstechnologie überschlagen sich die Entwicklungen, sobald technologische Errungenschaften neue Anwendungsfelder erschließen.

Etwa zeitgleich mit unserer Arbeit an NMP und der Erkenntnis, dass netzgestütztes Musizieren prinzipiell machbar ist, tauchten ab Mitte 2005 vergleichbare Projekte zunächst in der Forschung auf, um sehr schnell kommerzialisiert und bereits im nachfolgenden Jahr als Produkte beworben und vertrieben zu werden.

Die uns bekannten Versuche, diese Anwendung zu etablieren, sind bislang jedoch allesamt gescheitert, wie wir exemplarisch an den beiden folgenden Vertretern verdeutlichen wollen. Der erste ist dabei das bereits unter den verwandten Arbeiten vorgestellte Unternehmen eJamming, welches zum Projektstart von NMP Dienste für das instrumentelle Musizieren über MIDI angeboten hatte. Als Ergebnis von Beteiligungen an unterschiedlichen Forschungsprojekten wurde die Anwendung *eJamming AUDiiO* zur Teilnahme an Jam-Sessions im Internet entwickelt und 2006 auf der Musikmesse Frankfurt höchst publikumswirksam der Breiten Masse demonstriert.

An der damaligen Beta-Phase haben wir uns beteiligt und die Anwendung während einer befristeten Dauer erprobt. Dabei haben wir u.A. die Akzeptanz anhand der zugehörigen Foren und Bewertungsbeiträge verfolgt. Die zunächst nahezu euphorische Erwartungshaltung der interessierten Musiker ist dabei rapide der Realität gewichen, dass ein wie auf der Messe demonstriertes Musizieren mit der allgemein verfügbaren Zugangstechnologie nicht realisierbar ist. In den Foren machten sich die Teilnehmer Luft und brachten ihre Enttäuschung hervor, die nicht selten in Betrugsvorwürfen und Beschimpfungen mündete. Der Hersteller zog seine Konsequenzen daraus und passte die Mindestanforderungen für den Betrieb von *AUDiiO* und die Beschreibung der Features an. Statt 200 kbps wurden beim Übergang aus der Beta- in die Releasephase 500 kbps für Up- und Downlink gefordert, aus der minimal erreichbaren Latenz von 20 ms sind 50 ms geworden, inklusive des Hinweises, dass diese nur bei geeigneter Hardware- und Softwareausstattung zu erreichen ist.

Die endgültige Anforderungsliste ähnelt der, die wir für den Betrieb von NMP aufgestellt haben. Angesichts der systembedingten Limitierungen und vorgegebener technischer Schranken ist das wenig überraschend und belegt lediglich, dass auch die AUDiiO Software das theoretisch erreichbare Optimum erreicht hat. Analog dazu gilt, dass für die Verwendung von AUDiiO die Zugangstechnik noch nicht ausreicht und für das Zustande kommen von Online Jams viel Glück und Geduld erforderlich ist. Das Angebot von eJamming ist auch heute noch für ein pauschales Zugangsentgelt von knapp zehn US-Dollar pro Monat unter `http://www.ejamming.com/` zugänglich. Ein regelmäßiger Besuch der nahezu verwaisten Foren und Session Räume belegt bislang jedoch die Inaktivität und Unattraktivität des Angebots.

Der zweite exemplarische Anbieter ist das ebenfalls in den USA beheimatete Unternehmen *Lightspeed Audiolabs Inc.*, welches von namhaften Personen aus Forschung und IT-Industrie gegründet wurde und sich im Mai 2007 mit einem Wagniskapital von 1.25 Mio US$ aufmachte, die Musikwelt zu einem vernetzten Dorf zu machen. Nach einer kurzen Phase erwartungsvoller Versprechen ging der Dienst und die dazugehörige Anwendung *jamNow* ans Netz. Die beworbenen Fähigkeiten der Plattform klangen für den Musiker traumhaft, für uns Projektmitarbeiter am NMP dagegen wenig glaubhaft.

Es wurde eine Plug-and-Play Funktionalität – also Kopfhörer und Mikrofon an den PC anschließen und losmusizieren – geboten, entgegen unserer Erfahrung, dass eine akzeptable Latenz die selektive Auswahl von Audiohardware und -Schnittstelle erfordert. Gegen das Bewerben des eigens entwickelten proprietären Audiocodecs, der latenzfrei arbeitet und gleichzeitig sehr hohe Kompressionsraten und Audioqualität liefert, sprechen die in Abschnitt 6.4 beschriebenen prinzipiellen Faktoren. Ganz ohne Kenntnis der vorliegenden

Arbeit war die Ankündigung, mit jamNow seien Sitzungen weltweit möglich, als realitätsfern erkennbar, denn allein die Ausbreitungsverzögerung elektromagnetischer Wellen im globalen Maßstab übersteigt die für netzgestütztes Musizieren tolerierbaren Latenzen mehrfach.

Das Ergebnis dieser überoptimistischen Versprechen hat nicht lange auf sich warten lassen: die an der Testphase teilnehmenden Musiker überhäuften die Foren mit negativen Rückmeldungen darüber, dass der Dienst entweder überhaupt nicht funktionierte oder für das Musizieren gänzlich unbrauchbar war. Auch dieses Feedback haben wir als Teil der Akzeptanzstudie für NMP verfolgt und dabei festgestellt, dass knapp zwei Wochen nach Projektstart die Aktivität bei jamNow praktisch auf Null zurückgefallen ist. Weniger als ein Jahr nach Eröffnung verschwanden Mitte 2008 dann sowohl die Seite des Betreibers `http://www.lighspeedaudiolabs.com` als auch die des Dienstes `www.jamnow.com` aus dem Netz und sind heute nur noch aus dem Internet Archiv abrufbar.

Aus diesen gescheiterten Ansätzen können für die Zukunft und die Etablierung von NMP zwei wesentliche Lehren gezogen werden:

Zuverlässige Funktionalität

Wenn Musiker NMP als nützliches Instrument in ihren Alltag etablieren sollen, dann muss der Dienst jederzeit und mit gleich bleibenden Eigenschaften funktionieren. Eine Einsatzbereitschaft von 99 % ist nicht ausreichend, da angesichts des enormen Aufwandes für Organisation und Vorbereitung jeder Musiker nach der ersten verhinderten Sitzung NMP dauerhaft meiden wird. Ebenso strikt sind die Anforderungen hinsichtlich der Betriebsparameter: die Latenz darf zwischen den Sitzungen nur wenig schwanken und sollte im Idealfall immer konstant und unterhalb der anvisierten Schranke von 30 ms bleiben, während dabei gleichzeitig eine dauerhaft hohe Audioqualität erreicht werden muss. Unsere NMP Anwendung berücksichtigt diese Vorgabe und verweigert den Betrieb bis auf Widerruf, wenn eine Sitzung nicht mit zuverlässige Funktionalität durchgeführt werden kann.

Definierter Anwendungsradius

Die vollmundigen Versprechen der gescheiterten Ansätze führen zu einer nicht erfüllbaren Erwartungshaltung. Bei der Anwendung führt dies unweigerlich zu Enttäuschungen und Frust, der das weitere Interesse der Beteiligten an netzgestütztem Musizieren dauerhaft zerstört. Die Interessenten müssen daher vorab über den eingeschränkten potentiellen Anwendungsradius informiert werden, innerhalb dessen NMP als nützliches Werkzeug verwendbar ist. Dass dieser bereits durch die endliche Lichtgeschwindigkeit (bzw. der elektromagnetischen Ausbreitungsgeschwindigkeit) unverrückbar auf einen geografischen Radius von maximal 1500 km limitiert ist, vereinfacht die Argumentation für eine weitere Verringerung aufgrund von längeren Netzdistanzen und Verarbeitungszeiten in den Netzknoten. Der resultierende Anwendungsradius in der Größenordnung von 500 km Netzdistanz wurde von den an NMP beteiligten Musikern durchweg für ausreichend befunden, globales Musizieren wurde dagegen selten als Anwendungsfall genannt. Auch dieser Punkt wird von unserer NMPSoftware berücksichtigt, indem vor dem Aufbau von Sitzungen deren Netzradius geschätzt und geplante Sessions dahin gehend geprüft werden.

Angesichts dieser fundamentalen Anforderungen wird NMP ein kontinuierlicher schrittweiser Zugang zum Alltag, so wie es der Internet-Telefonie gegönnt war, verwehrt. Wir erinnern uns: ihren Anfang machte die IP-Telefonie zu Zeiten, in denen die Quasi-Monopolisten in der Kommunikationsbranche hohe Gebühren für Fern- und Auslandsgespräche diktieren konnten. Ausländische Studenten und Partner von Fernbeziehungen entdeckten die Möglichkeit, über diese neue Kommunikationsform enorme Kosten für den Kontakt mit Familie und Freunde einsparen zu können. Freilich war im frühen Stadium des Internets der Ausbau der Netzinfrastruktur nicht ausreichend, um eine dem Festnetz vergleichbare Leistung hinsichtlich Latenz, Au-

dioqualität und Zuverlässigkeit bieten zu können. Aber die Teilnehmer waren bereit und in der Lage, sich diesen Widrigkeiten anzupassen – und wenn diese zeitweise allzu hinderlich waren, konnte man ja immer noch auf das Telefon zurückgreifen. Als Teil der Anforderungen an das Netz, die seinen Ausbau angetrieben haben, war die IP-Telefonie wesentlich daran beteiligt, wechselseitig neue Leistungsanforderungen zu generieren und die Entwicklung voranzutreiben. Diese ist mittlerweile für die IP-Telefonie abgeschlossen: über das Internet geführten Gespräche haben die Telefonie hinsichtlich Audioqualität überholt und sind bei Zuverlässigkeit und Latenz gleichgezogen.

Für NMP als Wegbereiter einer neuen Anwendung gibt es dagegen keinen Fallback, auf den die Musiker bei Nichtgefallen zurückgreifen können. Damit überhaupt ein Einstieg in eine Phase wechselseitiger Anforderungen an die Netztechnik und umgesetzter technologischer Sprünge erfolgt, muss es bereits von einer kritischen Masse von Teilnehmern verwendet werden – was jedoch schon eine sehr solide Basisfunktionalität erfordert. Diese sehr hohe Hürde, die es auf einmal zu nehmen gilt, ist letztlich der Grund, warum das netzgestützte Musizieren noch immer das Labor nicht verlassen hat.

Als Schlusssatz wage ich in einem persönlichen Ausblick zu behaupten, dass diese Hürde innerhalb der kommenden fünf Jahre in vielen Regionen der Welt genommen werden wird. In meiner neuen Heimat Schweiz ist der Aufbau des Hochgeschwindigkeit Zugangsnetzes in Angriff genommen worden, in Ballungszentren wie Zürich oder Genf sind bereits 10 % der Haushalte über FTTH erschlossen, die für den Betreib von NMP mehr als ausreichend sind. Hier entfällt auch das Dreiecksproblem aufgrund des Peerings, da die geringe geografische Ausdehnung des Landes auch bei einem ungünstig platzierten Austauschknoten eine Netzdistanz innerhalb des Anwendungsradius sicherstellt. Ob das dann theoretisch machbare Musizieren im Netz auch praktisch zum Alltagswerkzeug wird, werde ich mit Spannung verfolgen. Eine passende Lösung steht mit dem von uns entwickelten Prototypen bereit, inklusive Klärung fundamentaler, in dieser Arbeit abgehandelter Fragestellungen.

LITERATURVERZEICHNIS

[1] GSM Recommendations 6.11. Substitution and Muting of lost Frames for Full Rate Speech Channels, September 1994.

[2] H.-M. Adler, P. Eitner, K. Ullmann, H. Waibel, and M. Wilhelm. Die netzinfrastruktur x-win des dfn. Technical report, DFN Verein, 2010.

[3] American National Standards Institute. Asymmetric digital subscriber line (ADSL) metallic interface. Technical Report ANSI T1.413 Issue 1, ANSI, 1995.

[4] American National Standards Institute. Objective video quality measurement using a Peak-Signal-to-Noise-Ratio (PSNR) full reference technique. Technical Report ANSI T1.TR.74-2001, ANSI, 2001.

[5] Apple. Darwin Streaming Server, February 2010. .

[6] S. Asbjørn and U.P. SVENSSON. A Low-Latency Full-Duplex Audio over IP Streamer. In *Linux Audio Developers' Conference*, volume 2006, pages 25–32, 2006.

[7] C. Audoin and B. Guinot. *The measurement of time*. Cambridge University Press New York, 2001.

[8] E. Berdahl, B. Verplank, J.O. Smith III, and G. Niemeyer. A physically intuitive haptic drumstick. In *Proc. Internat'l Computer Music Conf*, volume 1, pages 363–366, 2007.

[9] J. Blanchette and M. Summerfield. *C++ GUI programming with Qt 4*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2006.

[10] J.C. Bolot and A. Vega-Garcia. The case for FEC-based error control for packet audio in the Internet. *ACM Multimedia Systems*, 1997.

[11] Alexander Boltkov. Möglichkeiten und Grenzen für das gemeinsame Musizieren im Internet. Diplomarbeit, Institut für Betriebssysteme und Rechnerverbund, Technische Universität Braunschweig, July 2006.

[12] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed. RFC 3095 (Proposed Standard), July 2001. Updated by RFCs 3759, 4815.

[13] N. Bouillot. nJam user experiments: Enabling remote musical interaction from milliseconds to seconds. In *Proceedings of the 7th international Conference on New interfaces For Musical Expression*, pages 142–147. ACM, 2007.

[14] Nicolas Bouillot. The auditory consistency in distributed music performance: a conductor based synchronization. *ISDM Info et com Sciences for Decision*, 13(0):129–137, March 2004.

[15] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309 (Informational), April 1998.

[16] K. Brandenburg. MP3 and AAC explained. *Proceedings of AES 17th International Conference, XP008004053*, pages 99–110, 1999.

[17] S. Branigan, H. Burch, B. Cheswick, and F. Wojcik. What can you do with traceroute? *IEEE Internet Computing*, 5(5), September/October 2001.

[18] A. Carot, U. Kramer, and G. Schuller. Network Music Performance (NMP) in narrow band networks. *120th AES Convention*, May 2006.

[19] Chris Chafe, Michael Gurevich, Grace Leslie, and Sean Tyan. Effect of time delay on ensemble accuracy. In *Proceedings of the International Symposium on Musical Acoustics, (ISMA'04)*, pages 277–280, Nara, Japan, March/April 2004.

[20] Chris Chafe, Scott Wilson, Randal Leistikow, Dave Chisholm, and Gary Scavone. A simplified approach to high quality music and sound over IP. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFX'00)*, pages 159–164, Verona, Italy, December 2000.

[21] Jae Chung and M. Claypool. Analysis of active queue management. In *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on*, pages 359–366, April 2003.

[22] Jeremy R. Cooperstock and Stephen P. Spackman. The recording studio that spanned a continent. In *Proceedings of the IEEE International Conference on Web Delivering of Music (WEDELMUSIC)*, pages 161–169, Florence, Italy, November 2001.

[23] David Bryant. WavPack Hybrid Lossless Audio Compression, February 2010. .

[24] P. Davis et al. Jack audio connection kit. In *Linux Audio Developers' Conference*, volume 2003, 2003.

[25] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[26] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), August 1989. Updated by RFC 2236.

[27] Michael O'Brien EmbedThis. AppWeb embedded Web Server, February 2010. .

[28] Yoav Etsion, Dan Tsafrir, and Dror G. Feitelson. Effects of clock resolution on the scheduling of interactive and soft real-time processes. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 172–183, New York, NY, USA, 2003. ACM Press.

[29] D. Fober, Y. Orlarey, and S. Letz. Real Time Clock Skew Estimation over Network Delays. In *Proceedings of the International Music Conference, ICMA, San Francisco*. Citeseer, 2002.

[30] Marina Fomenkov, Ken Keys, David Moore, and K Claffy. Longitudinal study of internet traffic in 1998-2003. In *WISICT '04: Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin, 2004.

[31] André Frambach. Realisierung eines eingebetteten NMP Echtzeitclients und Implementierung einer konfigurationslosen Netzanbindung. Studienarbeit, Institut für Betriebssysteme und Rechnerverbund, Technische Universität Braunschweig, August 2007.

[32] D.J. Goodman and O.J. Wasem. Waveform substitution techniques for recovering missing speech segments in packet voice communications. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(6):1440–48, December 1986.

[33] J G Gruber and L Strawczynski. Subjective effects of variable delay and clipping in dynamically managed voice systems. *IEEE Transactions on Communications*, 33:801–808, 1985.

[34] X. Gu, M. Dick, Z. Kurtisi, U. Noyer, and L. Wolf. Network-centric Music Performance: Practice and Experiments. *IEEE Communications Magazine*, 43(6):86–93, jun 2005.

[35] B. Guinot. Is the International Atomic Time TAI a coordinate time or a proper time? *Celestial Mechanics and Dynamical Astronomy*, 38(2):155–161, 1986.

[36] V. Hardman, M.A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. In *Proceedings of INET*, volume 95, pages 171–178, 1995.

[37] J. Heckroth. A Tutorial on MIDI and Wavetable Music Synthesis. *Crystal Semiconductor Corp*, 1993.

[38] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer-Verlag, 1997.

[39] H. Holbrook, B. Cain, and B. Haberman. Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard), August 2006.

[40] IEEE. *IEEE 802.11-1999 Standard: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society, September 1999.

[41] RFC IETF. RTP Profile for Audio and Video Conferences with Minimal Control, January 1996.

[42] International Organization for Standardization and European Computer Manufacturers Association. Information processing: volume and file structure of CD-ROM for information interchange. International standard; ISO 9660 ISO 9660: 1988 (E), International Organization for Standardization, Geneva, Switzerland, April 1988.

[43] International Telecommunication Union. G.726: 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM). Rec. G.726, ITU-T, December 1990.

[44] International Telecommunication Union. Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s. Recommendation G.723.1, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, March 1996.

[45] International Telecommunication Union. Methods for the Subjective Assessment of small Impairments in Audio Systems including Multichannel Sound Systems. Rec. BS.1116-1, ITU-R, 1997.

[46] International Telecommunication Union. Method for Objective Measurements of Perceived Audio Quality. Rec. BS.1387, ITU-R, November 2001.

[47] International Telecommunication Union. Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. Rec. P.862, ITU-T, February 2001.

[48] International Telecommunication Union. G.114: One-way transmission time. Rec. G.114, ITU-T, May 2003.

[49] N. Jayant and S. Christensen. Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure. *Communications, IEEE Transactions on [legacy, pre-1988]*, 29(2):101–109, 1981.

[50] Joerg Bitzer and Tobias May and Zefir Kurtisi. Distant Teaching of Chamber Music via Local Area Networks, Poster presentation. In *120th Audio Engineering Society Convention, Pro Audio Expo*, Paris, France, May 2006.

[51] Hans J. Sitte John C. Eidson, Jogesh Warrior. Distributed control system architecture based on synchronized clocks, 1997. United States Patent US 6.654.356, 25.11.2003.

[52] JD Johnston and AJ Ferreira. Sum-difference stereo transform coding. *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, 2, 1992.

[53] AM Kondoz, J. Wiley, et al. *Digital Speech Coding for Low Bit Rate Communication Systems*. Wiley, 2004.

[54] Dimitri Konstantinas. Overview of a telepresence environment for distributed musical rehearsals. In *Proceedings of the ACM Symposium on Applied Computing (SAC'98)*, pages 456–457, Atlanta, USA, February 1998.

[55] Fernando Lindner Ramos, Marcio de Oliveira Costa, and Jônatas Manzolli. Virtual studio: distributed musical instruments on the web. In *Proceedings of the IXth Brazilian Symposium on Computer Music*, Campinas, Brazil, August 2003.

[56] Live Networks Inc., Ross Finlayson. LIVE555 Streaming Media Library, February 2010. .

[57] M. Lutzky, G. Schuller, M. Gayer, U. Kramer, and S. Wabnik. A guideline to audio codec delay. *AES 116th convention, Berlin, Germany*, pages 8–11, 2004.

[58] G.A. Miller and JCR Licklider. The intelligibility of interrupted speech. *The Journal of the Acoustical Society of America*, 22:167, 1950.

[59] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), March 1992.

[60] D. Mills. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC 4330 (Informational), January 2006.

[61] D.L. Mills. Network Time Protocol (NTP). RFC 958, September 1985. Obsoleted by RFCs 1059, 1119, 1305.

[62] D.L. Mills. Measured performance of the Network Time Protocol in the Internet system. RFC 1128, October 1989.

[63] N. Mitra and Y. Lafon. SOAP version 1.2 Part 0: Primer. *W3C Recommendation*, 24, 2003.

[64] Robert A. Moog. Midi: Musical instrument digital interface. *J. Audio Eng. Soc*, 34(5):394–404, 1986.

[65] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998. Updated by RFCs 3168, 3260.

[66] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 535–544, June 2002.

[67] BW Parkinson and JJ Spilker. Global positioning system: theory and practice. *Volumes I and II, American Institute of Aeronautics and Astronautics Inc., Washington, DC*, 1996.

[68] D. Phillips. A user's guide to ALSA. *Linux Journal*, 2005(136):3, 2005.

[69] J.O. Pickles. *An introduction to the physiology of hearing*. Academic Press, 2008.

[70] D. Piester, P. Hetzel, and A. Bauch. Zeit-und Normalfrequenzverbreitung mit DCF77. *PTB-Mitteilungen*, 114(4), 2004.

[71] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[72] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[73] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[74] R. Rahmani and O. Popov. Adaptive active queue management in heterogeneous networks. In *Information Technology Interfaces, 2004. 26th International Conference on*, pages 633–636, June 2004.

[75] F.L. Ramos, M. de Oliveira Costa, and J. Manzolli. Virtual studio: distributed musical instruments on the web. In *Proc. of IX Brazilian Symposium on Computer Music, Campinas, Brazil*, August 2003.

[76] Seon-Min Rhee, R. Ziegler, Jiyoung Park, M. Naef, M. Gross, and Myoung-Hee Kim. Low-cost telepresence for collaborative virtual environments. In *Visualization and Computer Graphics, IEEE Transactions on*, volume 13, pages 156–166, Research Triangle Park, NC, USA, January/February 2007.

[77] R. F. Rice. Some practical universal noiseless coding techniques. Technical Report JPL Publication 79-22, Jet Propulsion Laboratory, Pasadena, California, 1979.

[78] J. Rosenberg and H. Schulzrinne. RFC2733: An RTP Payload Format for Generic Forward Error Correction. *RFC Editor United States*, 1999.

[79] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320.

[80] David Salomon. *Data Compression: The Complete Reference*. Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 2007. With contributions by Giovanni Motta and David Bryant.

[81] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod. A new technique for audio packet loss concealment. *GLOBECOM-NEW YORK-*, pages 48–52, 1996.

[82] G.P. Scavone. RtAudio: A cross-platform C++ class for realtime audio input/output. In *Proc. 2002 Int. Computer Music Conf*, pages 196–199. Citeseer, 2002.

[83] Nathan Schuett. *The Effects of Latency on Ensemble Performance*. PhD thesis, Stanford University, May 2002.

[84] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[85] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard), April 1998.

[86] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. RFC 2212 (Proposed Standard), September 1997.

[87] J. Smart, R. Roebling, V. Zeitlin, R. Dunn, et al. wxWidgets 2.6.3: A portable C++ and Python GUI toolkit, March 2006. .

[88] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 1831 (Proposed Standard), August 1995.

[89] Steinberg. ASIO 2.0 Development Kit: Audio Streaming Input Output Specification. Documentation, Steinberg, 2005.

[90] A. Tannenbaum. Moderne Betriebssysteme, Studienbücher der Informatik, 1995.

[91] B. Teitelbaum, J. Sikora, and T. Hanss. Quality of service for Internet2. In *First Internet2 Joint Applications/Engineering QoS Workshop*, page 5. Citeseer, 1998.

[92] T. Thiede, WC Treurniet, R. Bitto, C. Schmidmer, T. Sporer, JG Beerends, M. Keyhl, C. Colomes, M. Lever, G. Stoll, et al. PEAQ-der künftige ITU-Standard zur objektiven Messung der wahrgenommenen Audioqualität. *Proceedings of the 20th International Convention on Sound Design (Tonmeistertagung), Verlag KG Saur, München*, pages 724–766, 1999.

[93] R.M. Warren. *Auditory perception: A new analysis and synthesis*. Cambridge university press, 1999.

[94] O.J. Wasem, D.J. Goodman, C.A. Dvorak, and H.G. Page. The effect of waveform substitution on the quality of PCM packet communications. *IEEE Transactions on Acoustics Speech and Signal Processing*, 36(3):342–348, 1988.

[95] G. Wong et al. Speed of sound in standard air. *Journal of the Acoustical Society of America*, 79(5):1359–1366, 1986.

[96] J. Wroclawski. The Use of RSVP with IETF Integrated Services. RFC 2210 (Proposed Standard), September 1997.

[97] Xiph Org., Josh Coalson. Free Lossless Audio Codec (FLAC), February 2010. .

[98] Aoxiang Xu, Wieslaw Woszczyk, Zack Settel, Bruce Pennycook, Robert Rowe, Philip Galanter, Jeffrey Bary, Geoff Martin, Jason Corey, and Jeremy R. Copperstock. Real-Time Streaming of Multichannel Audio Data over Internet. *Journal of the Audio Engineering Society*, 48(7-8):627–641, July/August 2000.